The Age of Programmable Networks

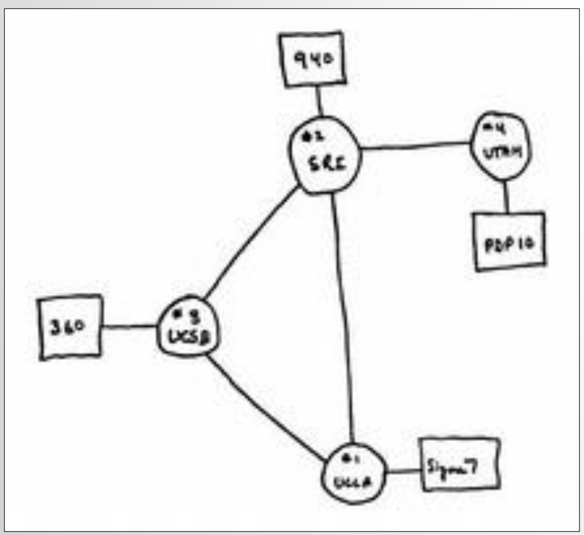
Algorithms for Designing Flexible and Robust SDNs

Stefan Schmid

Aalborg University, DK & TU Berlin, DE

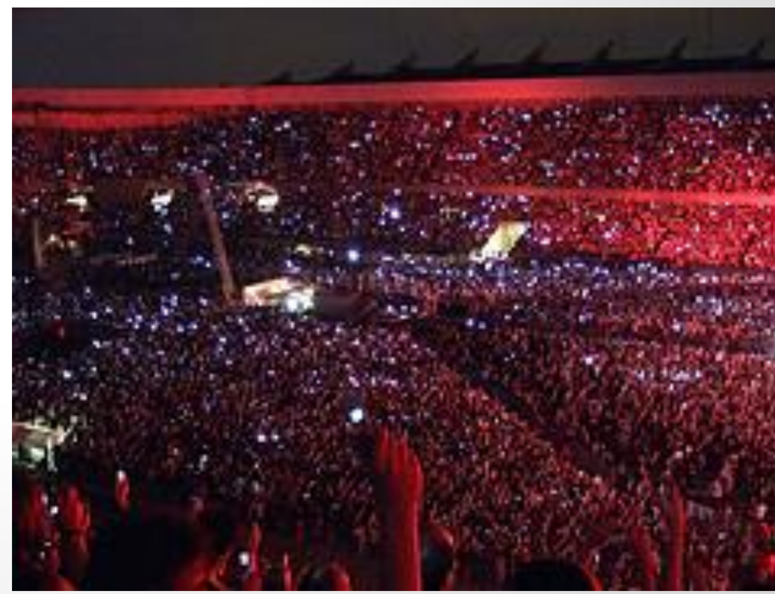
Why SDN? Why RNDM? The Internet Works!

- Datacenter networks, enterprise networks, Internet: a critical infrastructure of the information society
- While many Internet protocols hardly changed…
- … we have seen a huge shift in scale and application diversity



Goal: connectivity between researchers

Applications: file transfer, email



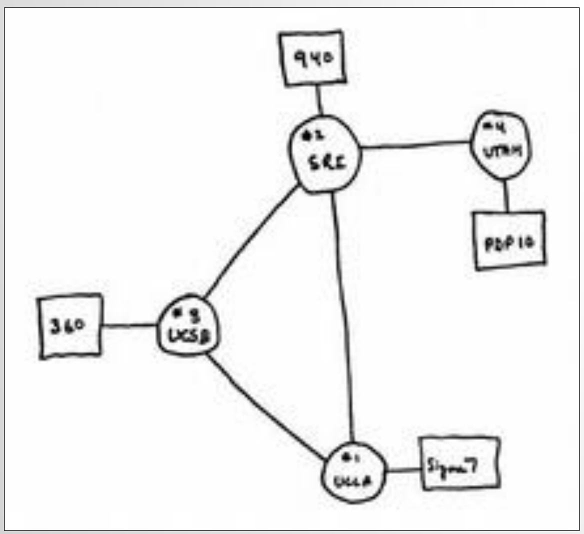
Goal: QoS, security, …

Applications: live streaming, IoT, etc.

Why SDN? Why RNDM? The Internet Works!

- Datacenter networks, enterprise networks, Internet: a critical infrastructure of the information societ
- While many Internet protocols hardly c
- … we have seen a huge shift in scale and application diversity

Hardly any outages over the last decades!



Goal: connectivity between researchers

Applications: file transfer, email

Goal: QoS, security, …

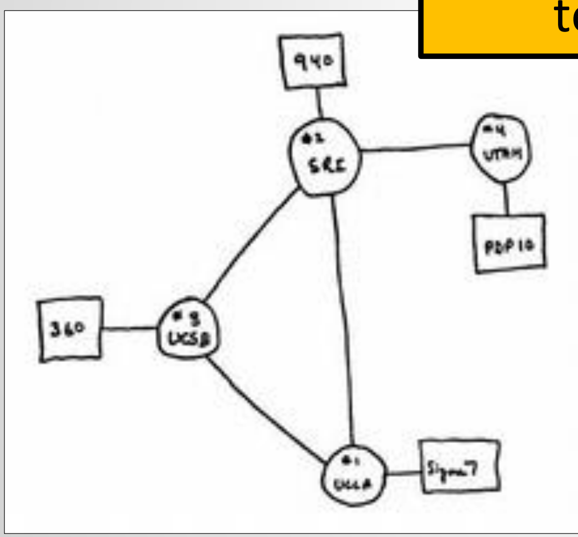
Applications: live streaming, IoT, etc.

Why SDN? Why RNDM? The Internet Works!

- Datacenter networks, enterprise networks, Internet: a critical infrastructure of the information societ
- While many Internet protocols hardly c
- … we have seen a huge shift in scale and application diversity

Hardly any outages over the last decades!

But new requirements and challenges, especially in terms of reliability!



Goal: connectivity between researchers

Applications: file transfer, email

Goal: QoS, security, …

Applications: live streaming, IoT, etc.

Reliability is The Big Networking Challenge

Even techsavvy companies struggle to provide reliable operations

Mostly manual and ad-hoc!

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

Source: Talk by Nate Foster at DSDN Workshop

Reliability Challenge 1: Lack of Good Debugging Tools

The Wall Street Bank Anecdote

- Outage of a data center of a Wall Street investment bank: lost revenue measured in USD 10^6 / min!

- Quickly, assembled emergency team:

The compute team: quickly 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.

The networking team: All the networking team had were two tools invented over twenty years ago [*ping* and *traceroute*] to merely test end-to-end connectivity. Neither tool could reveal problems with the switches, the congestion experienced by individual packets, or provide any means to create experiments to identify, quarantine and resolve the problem.

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

Reliability Challenge 1: Lack of Good Debugging Tools

The Wall Street Bank Anecdote

- Outage of a data center of a Wall Street investment bank: lost revenue measured in USD 10^6 / min!

Guess who gets blamed?

- Quickly, assembled emergency team:

The compute team: quickly 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.

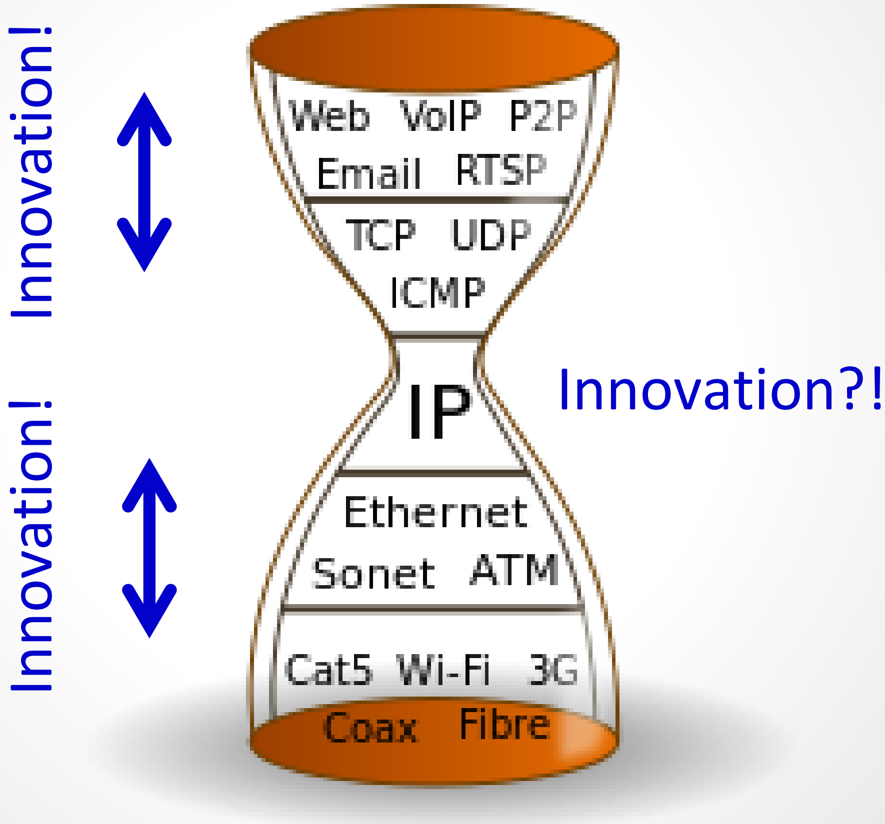
The storage team: similarly equipped, showing which file system logs were affected, and already progressing with workaround programs.

The networking team: All the networking team had were two tools invented over twenty years ago [*ping* and *traceroute*] to merely test end-to-end connectivity. Neither tool could reveal problems with the switches, the congestion experienced by individual packets, or provide any means to create experiments to identify, quarantine and resolve the problem.

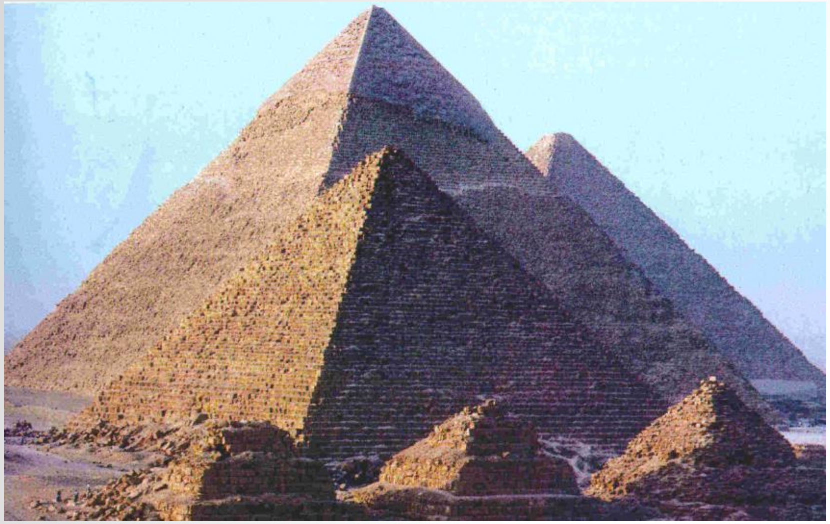
Source: «The world's fastest and most programmable networks»
White Paper Barefoot Networks

Reliability Challenge 2: Lack of Innovations

Have a good idea for a new reliable network protocol? Forget it!



Reliability Challenge 3: Security

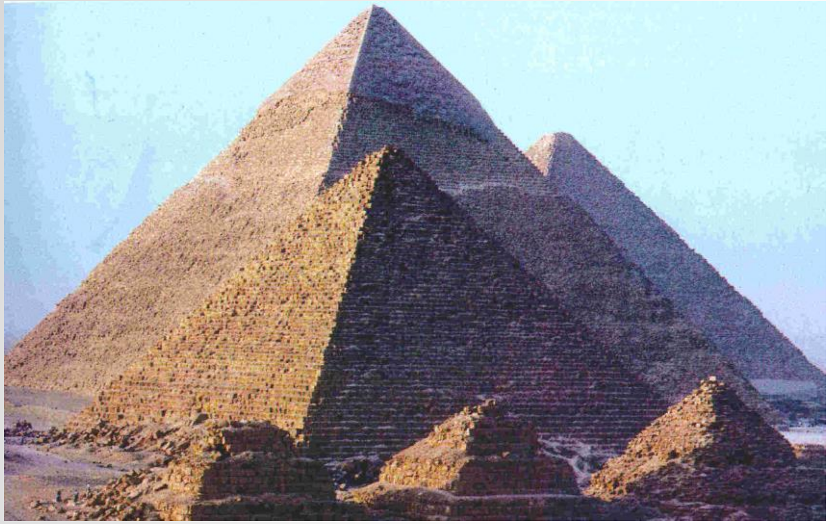


The Internet on first sight:

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

Source: Slide by Adrian Perrig

Reliability Challenge 3: Security



The Internet on first sight:
- Monumental
- Passed the "Test-of-Time"
- Should not and cannot be changed

The Internet on second sight:
- Antique
- Britle
- Successful attacks more and more frequent (e.g., based on IoT)

Source: Slide by Adrian Perrig

Reliability Challenge 3: Security

Security / #CyberSecurity

SEP 25, 2016 @ 10:00 AM 41,011 VIEWS

How Hacked Cameras Are Helping Launch The Biggest Attacks The Internet Has Ever Seen

Thomas Fox-Brewster, FORBES STAFF

I cover crime, privacy and security in digital and physical forms. **FULL BIO**

Recent "Attack of the (Internet-)Things"
(aka babyphone attack)

Reliability Challenge 3: Security

Remark: Assumptions have changed!

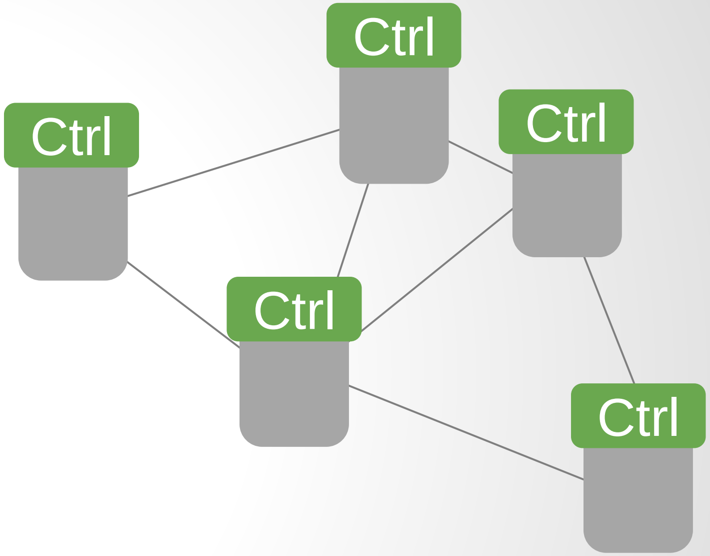
Danny Hillis, TED* talk, Feb. 2013, about trust in the Internet in the 80s: “There were two Dannys. I knew both. Not everyone knew everyone, but there was an atmosphere of trust.”

The paper by David Clark about “The Design Philosophy of the DARPA Internet Protocols” does not even contain the term security.

A Promising Trend:
Programmable Networks
aka *Software-Defined Networks (SDNs)*

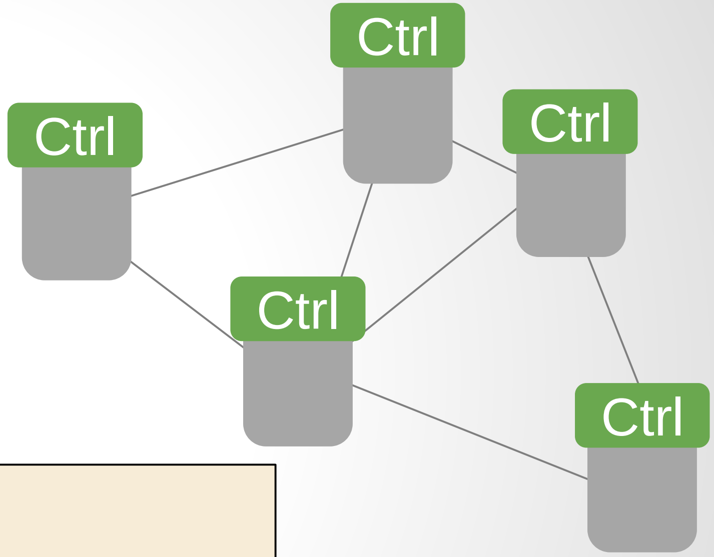
SDN: What is it about?

Traditional networks:
standardized and fixed
distributed algorithms

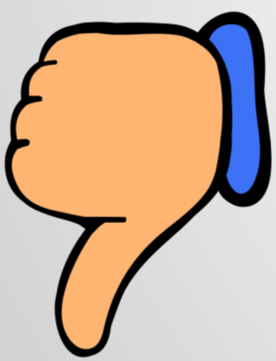


SDN: What is it about?

Blackbox, complex distributed configuration.



Traditional networks: standardized and fixed distributed algorithms



Drawbacks, e.g.:

- fixed algorithms
- blackbox devices
- slow reconvergence after failures («bad news slow»)
- complex and hard to debug

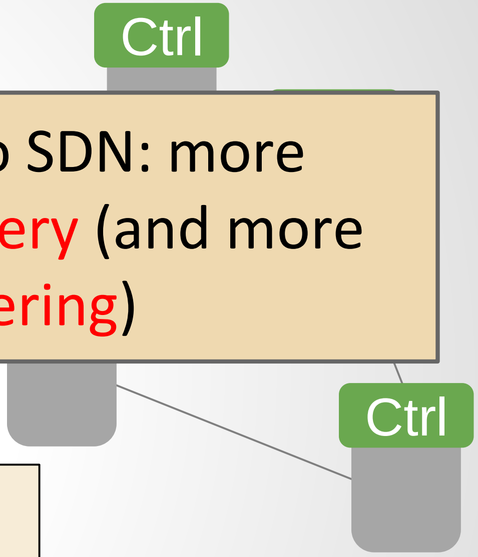
SDN: What is it about?

Traditional networks:

standa

distribu

Key reasons for Google's move to SDN: more predictable and faster failure recovery (and more fine-grained traffic engineering)



acks, e.g.:

algorithms

box devices

reconvergence after failures («bad news slow»)

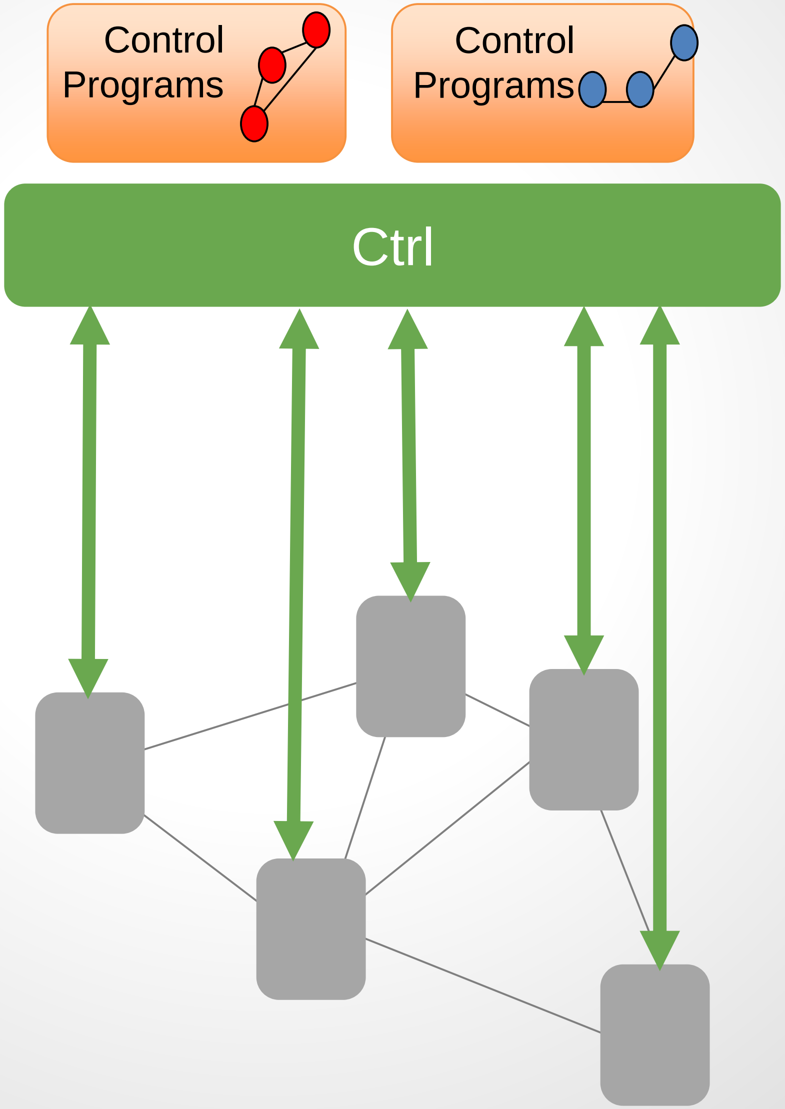
A Purpose-Built Global Network: Google's Move to SDN

- complex and hard to debug

SDN = Logical Centralization

In a nutshell:

SDN outsources and consolidates control over multiple devices to (logically) centralized software controller

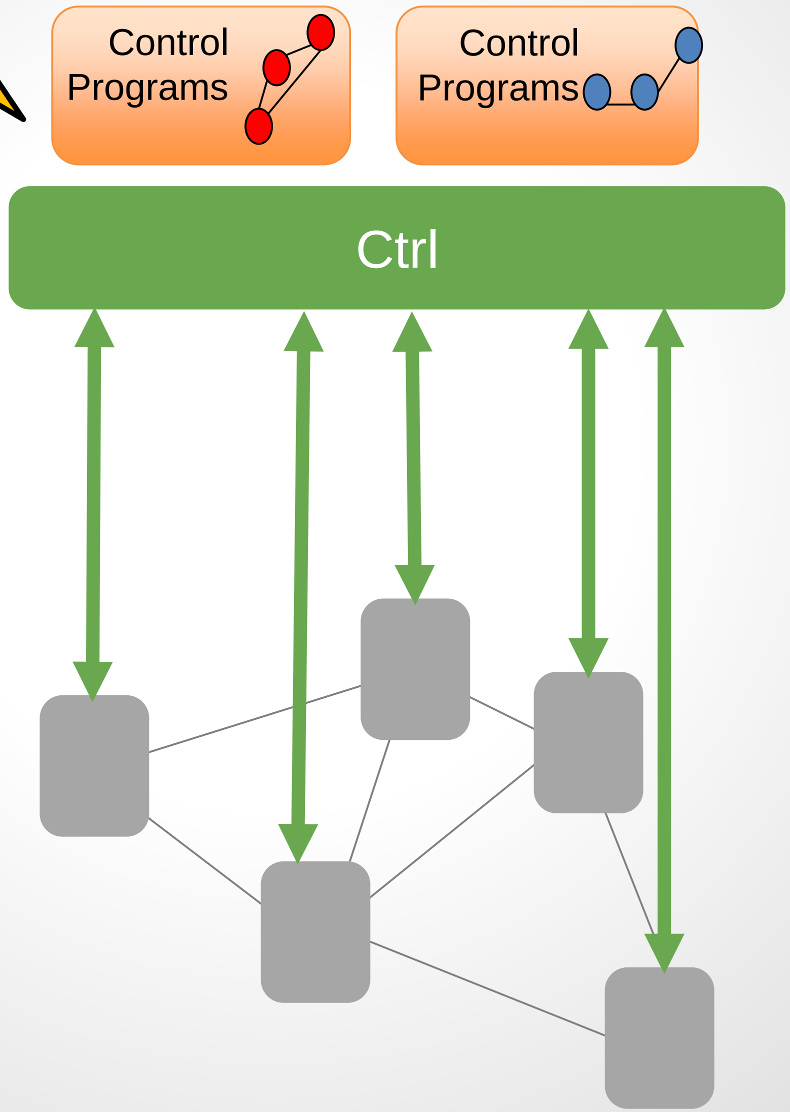


SDN = Logical Centralization

Algorithms run on server in software here: **Your RNDM Algorithm**!

In a nutshell:

SDN outsources and consolidates control over multiple devices to (logically) centralized software controller

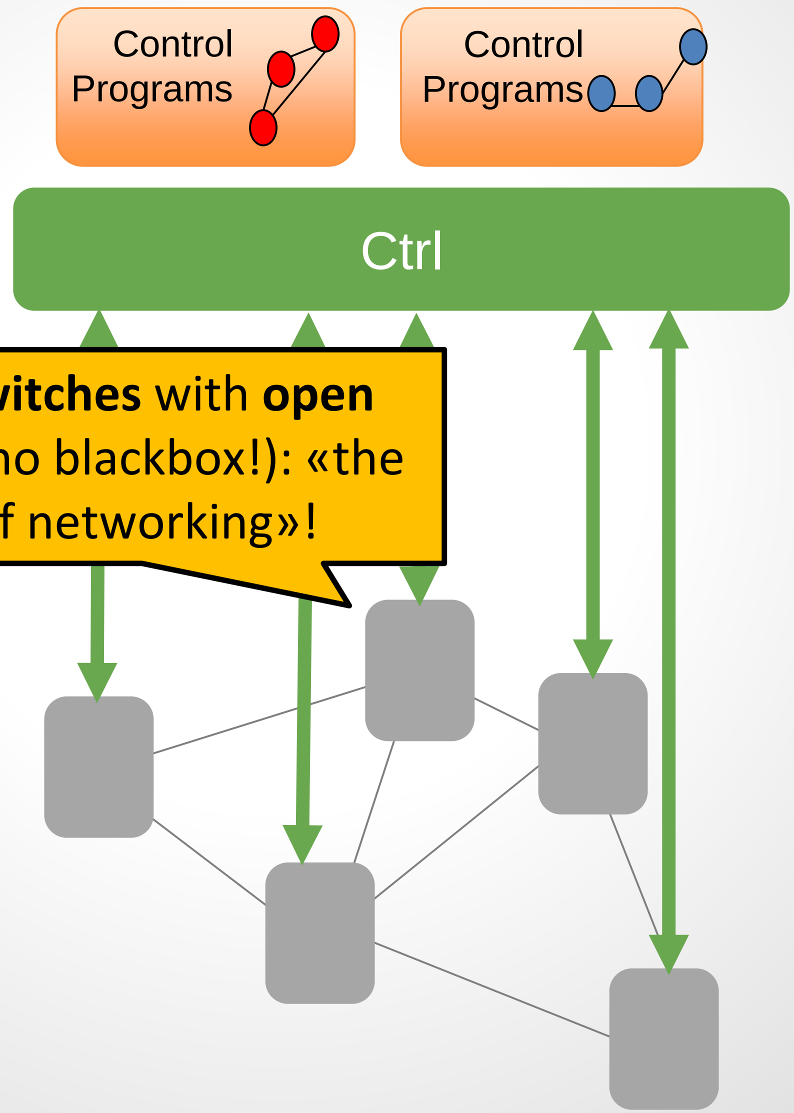


SDN = Logical Centralization

In a nutshell:

SDN outsources and consolidates c
over multiple
devices to (log
centralized software
controller

Simple switches with **open interface** (no blackbox!): «the Linux of networking»!



SDN = Logical Centralization

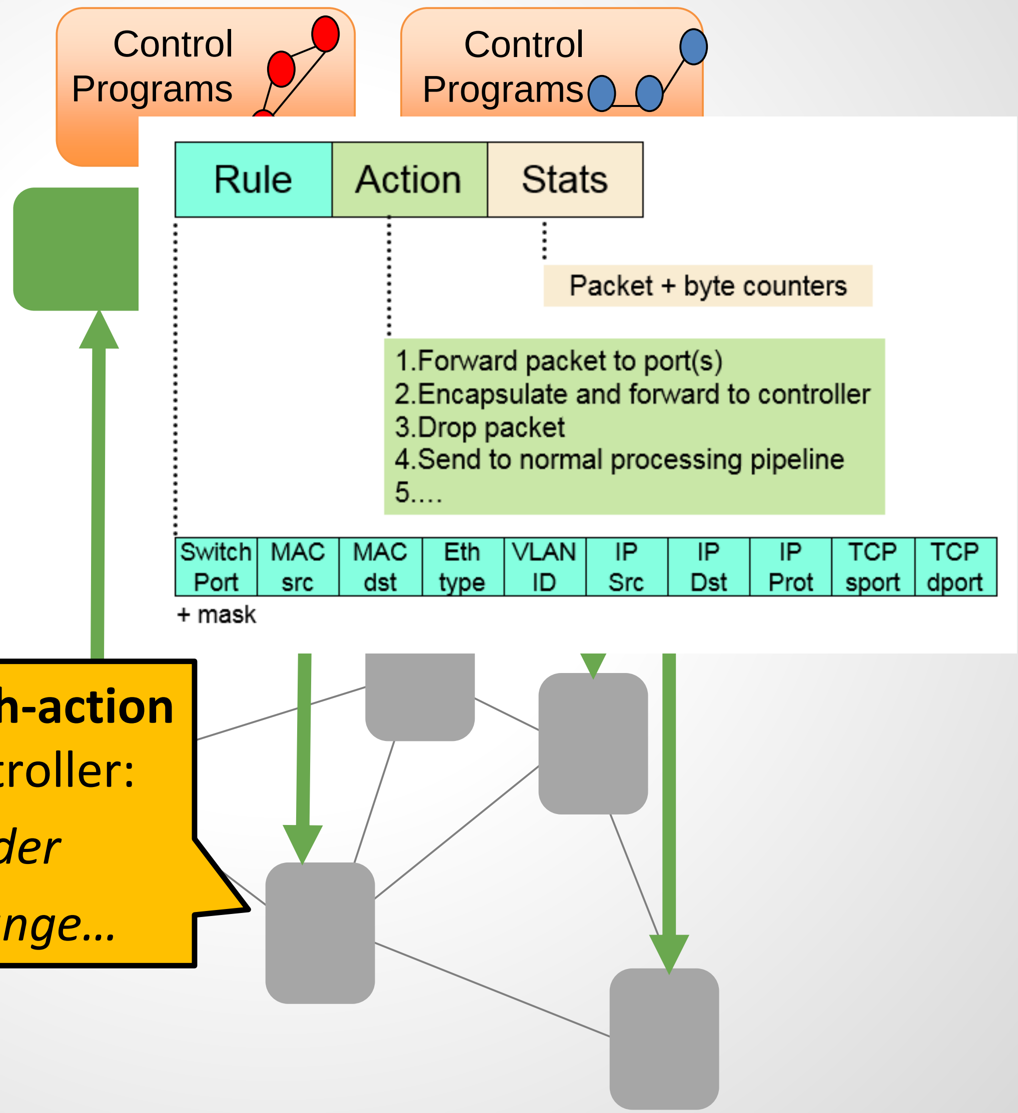
In a nutshell:

SDN outsources and consolidates control over multiple devices to (logically) centralized software cont

Concretely: set of **match-action rules** installed by controller:

match packet header

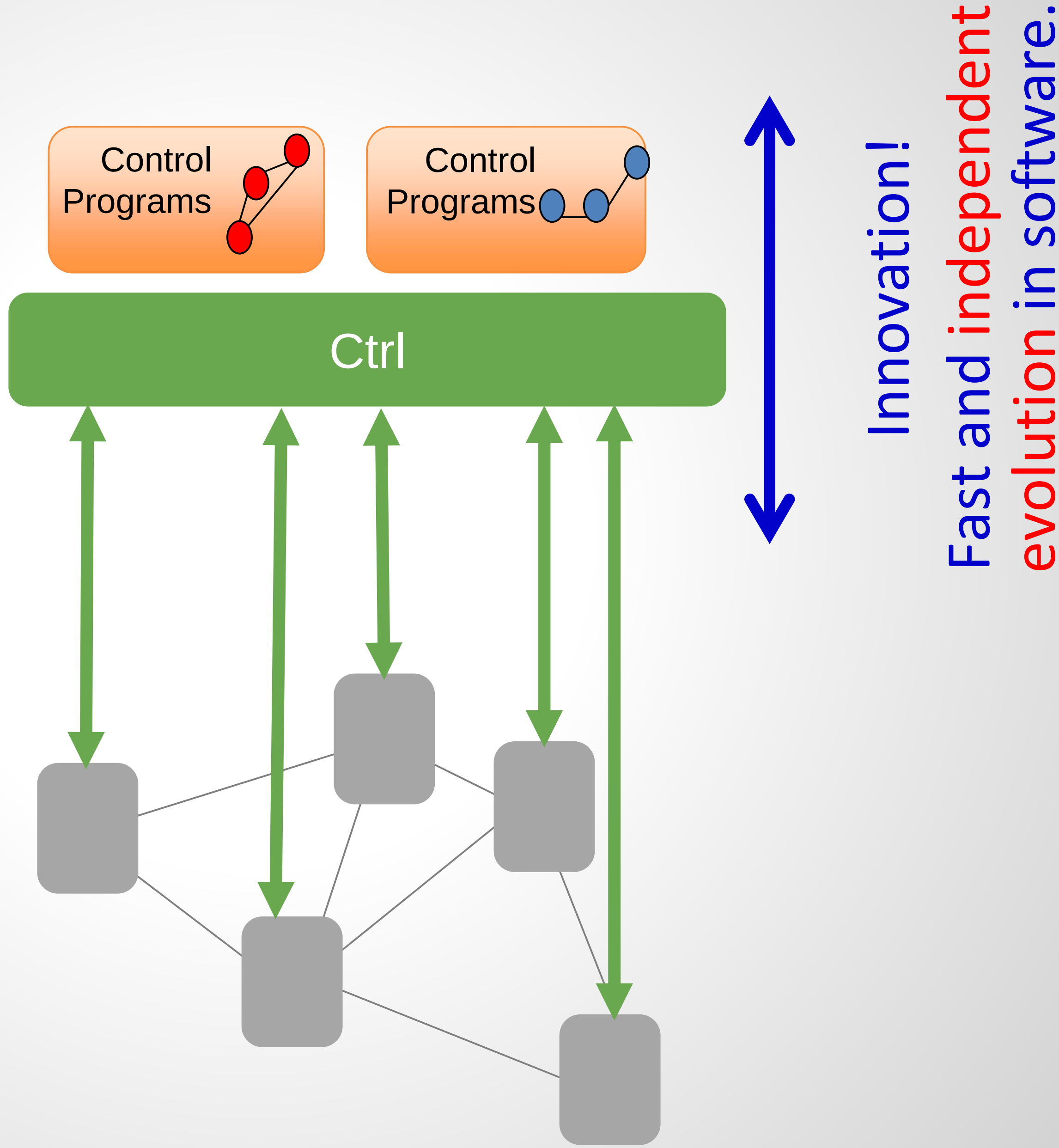
=> forward/drop/change…



SDN = Logical Centralization

In a nutshell:

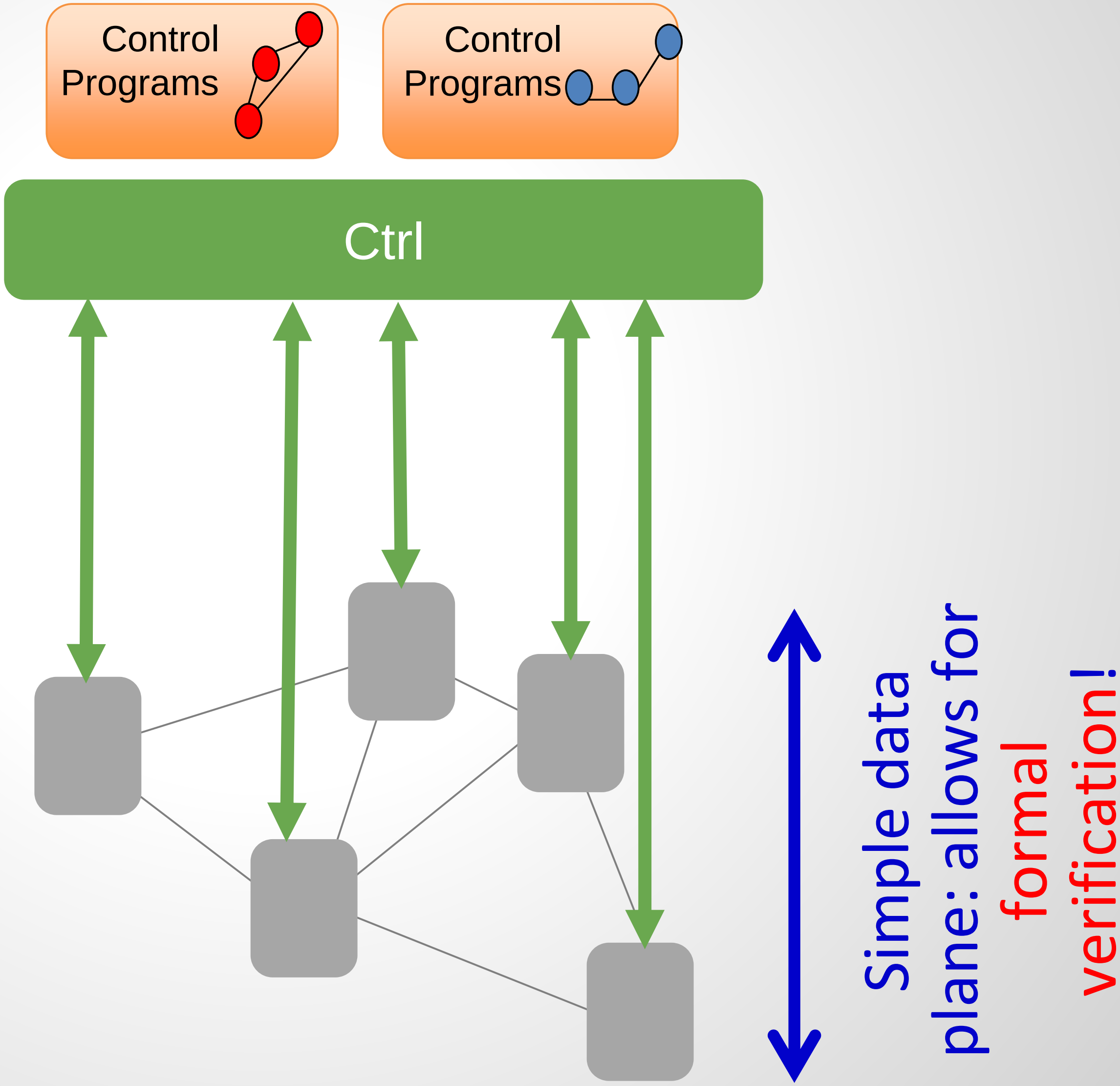
SDN outsources and consolidates control over multiple devices to (logically) centralized software controller



SDN = Logical Centralization

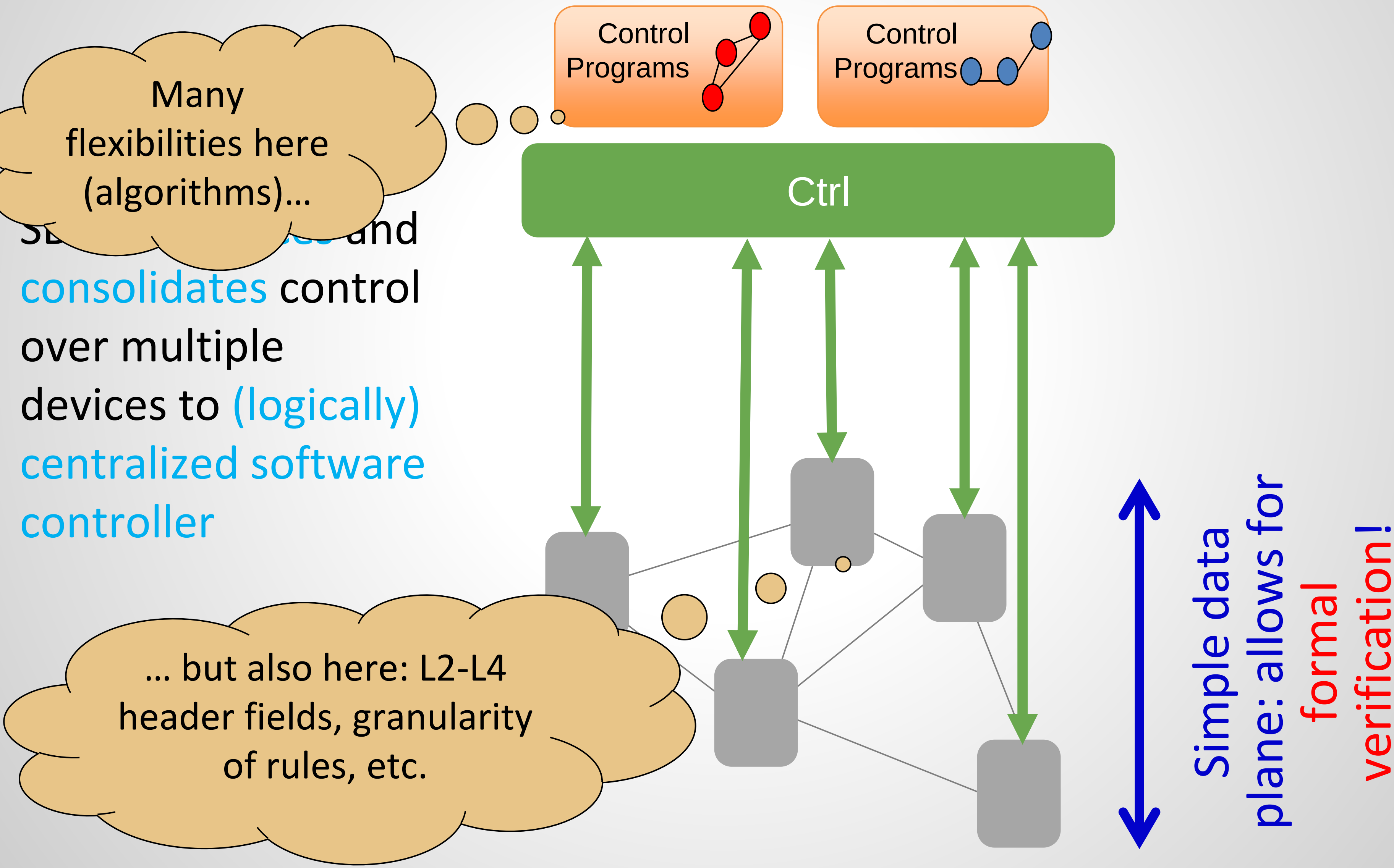
In a nutshell:

SDN outsources and consolidates control over multiple devices to (logically) centralized software controller



SDN = Logical Centralization

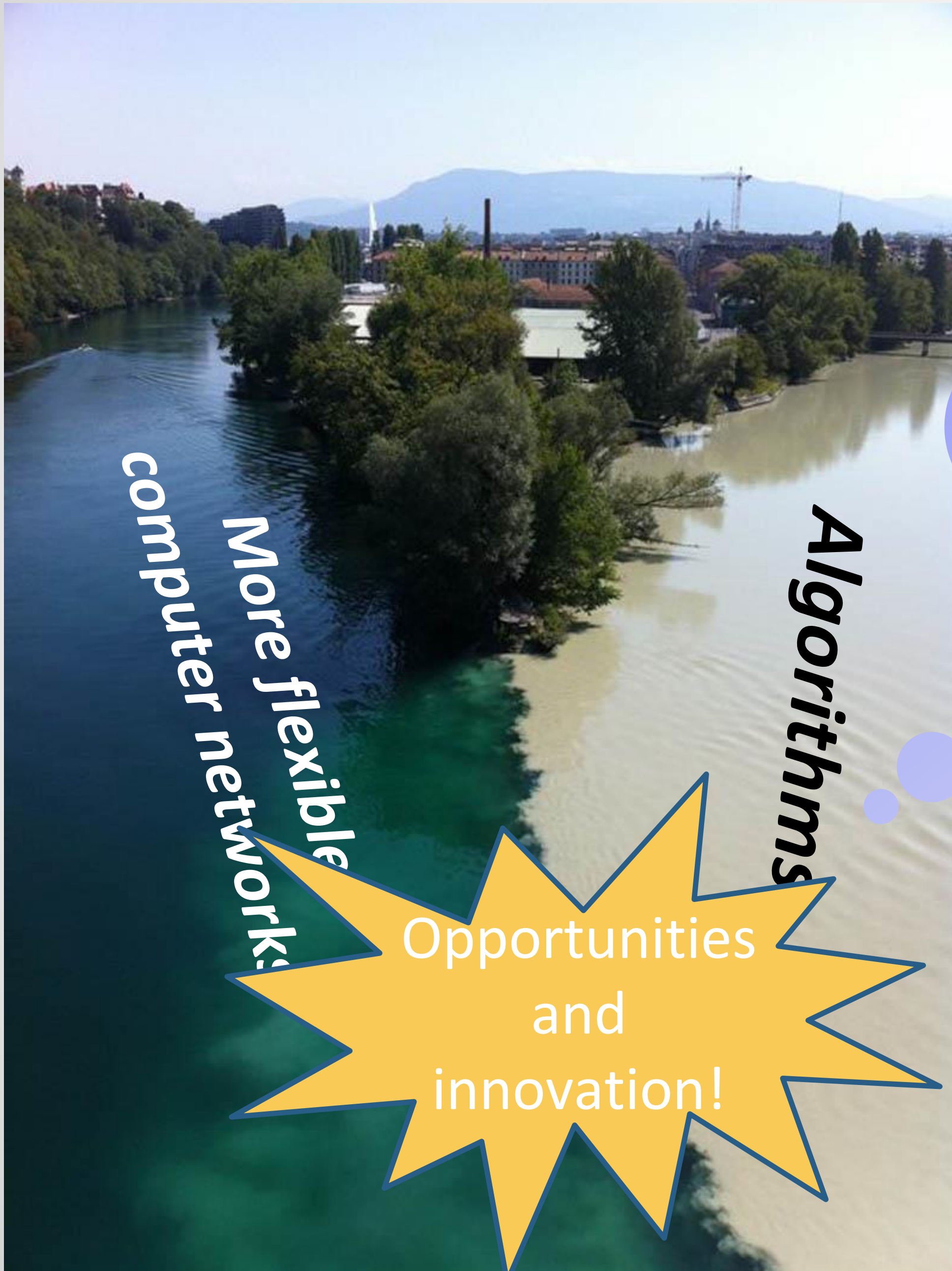
SD… …es and consolidates control over multiple devices to (logically) centralized software controller



A rehash: It's a great time to be a scientist!

"We are at an interesting inflection point!"
Keynote by George Varghese at SIGCOMM 2014

A rehash: It's a great time to be a scientist!



But how to exploit
these flexiblities?
How not to shoot
in our feet?
New RNDM
challenges!

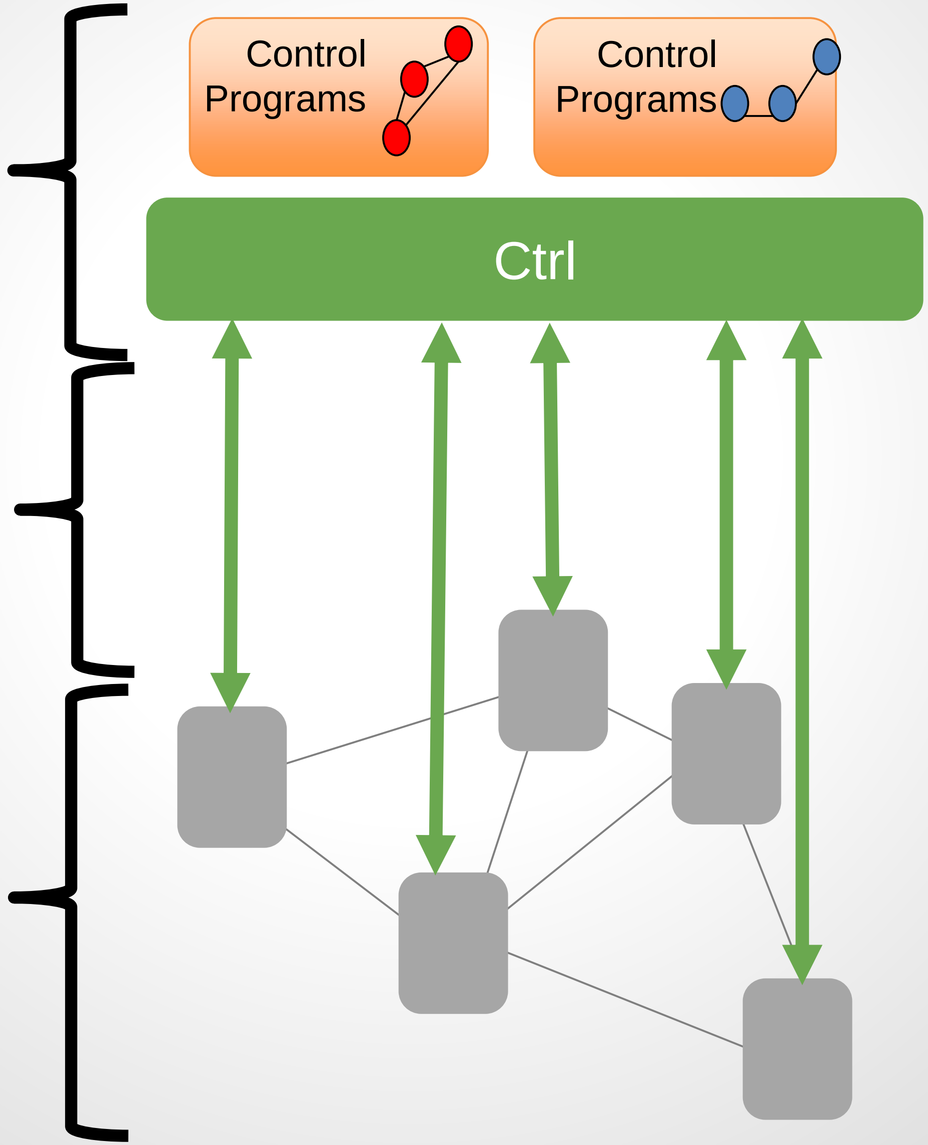
"We are at an interesting
inflection point!"
Keynote by George Varghese
at SIGCOMM 2014

New RNDM Challenges on Several Fronts

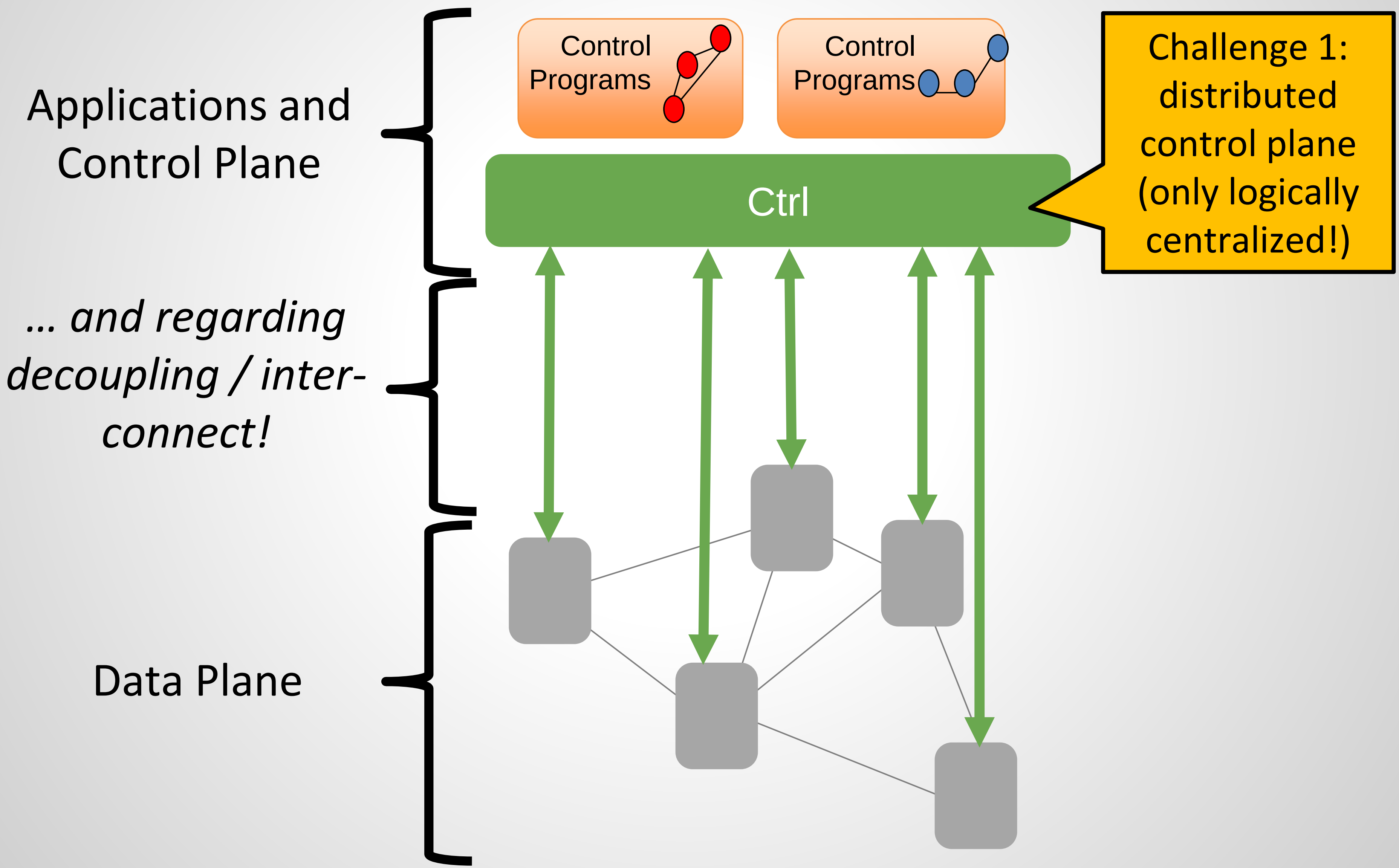
Applications and Control Plane

... and regarding decoupling / inter-connect!

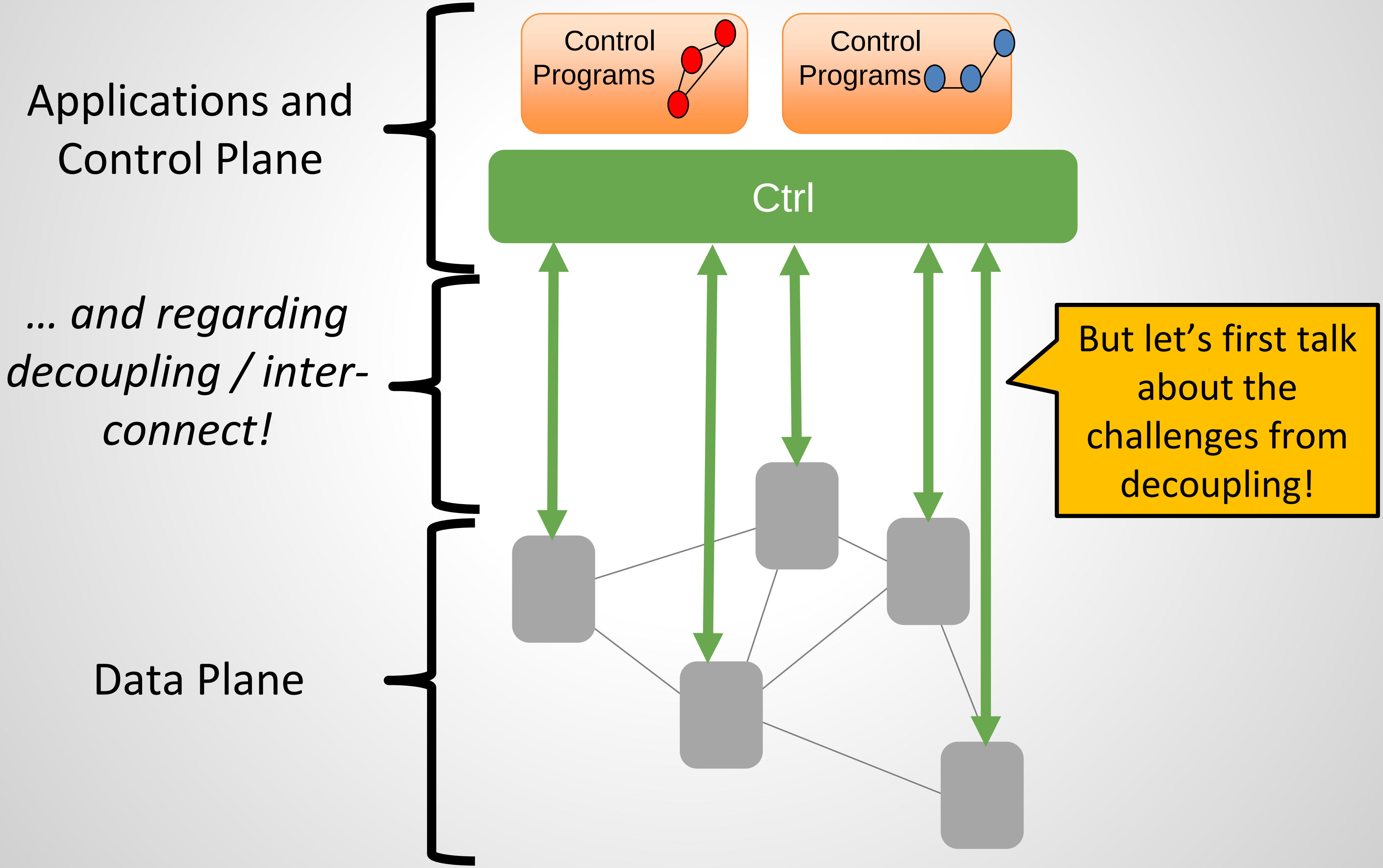
Data Plane



New RNDM Challenges on Several Fronts



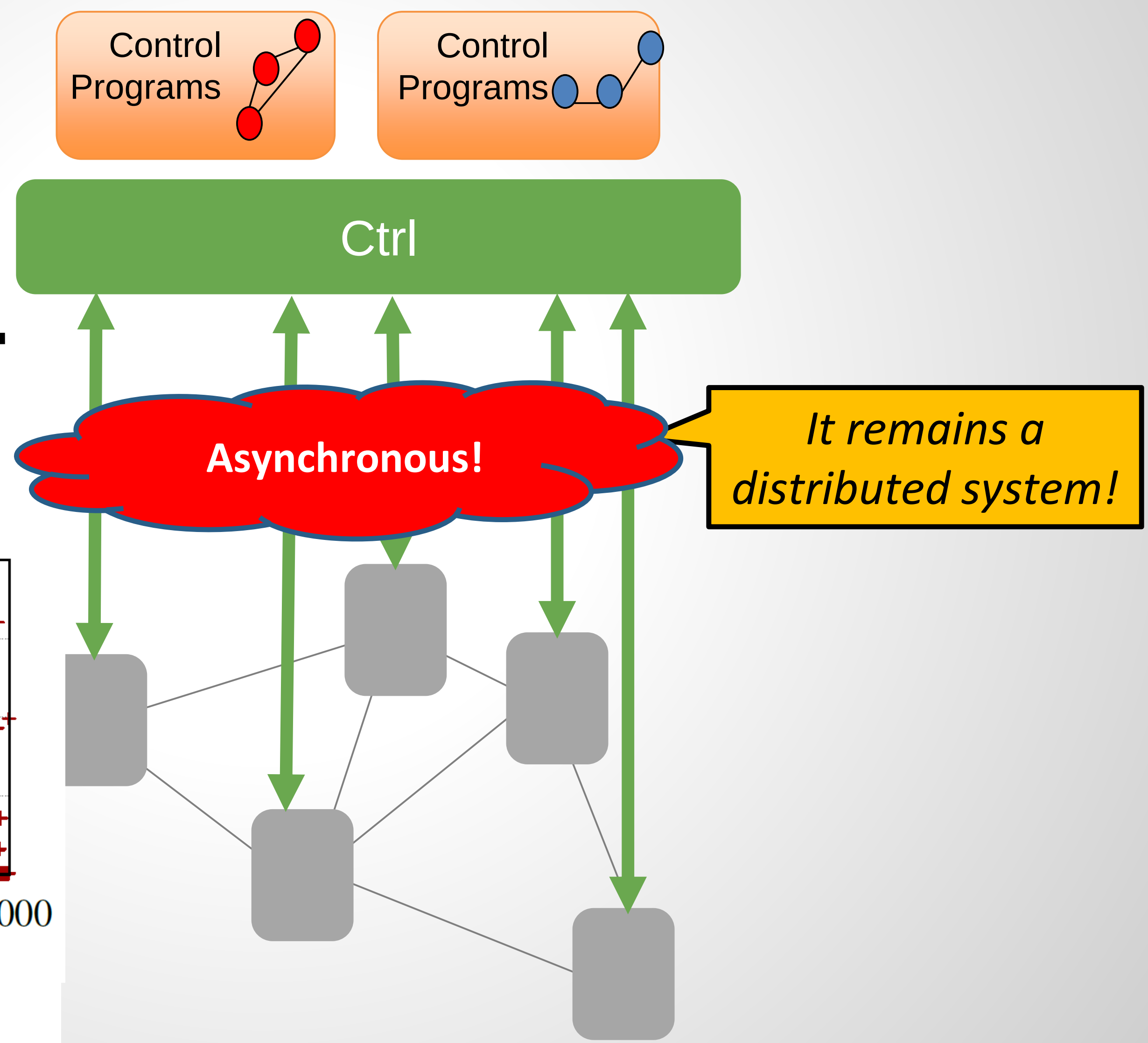
New RNDM Challenges on Several Fronts

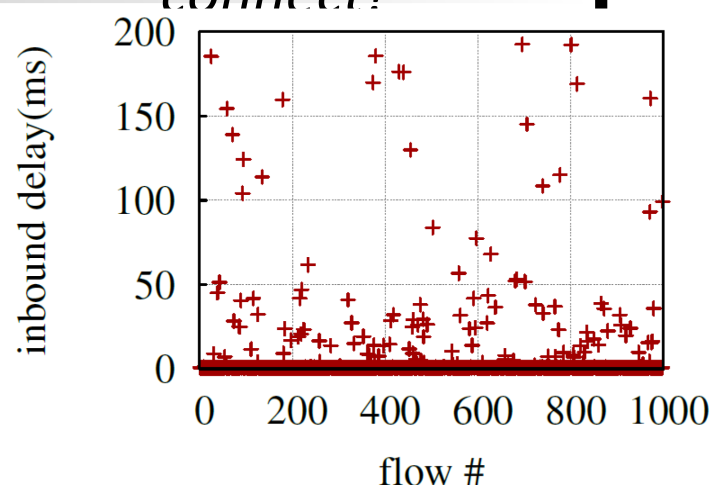


New RNDM Challenges on Several Fronts

Applications and Control Plane

… and regarding decoupling / inter-connect!

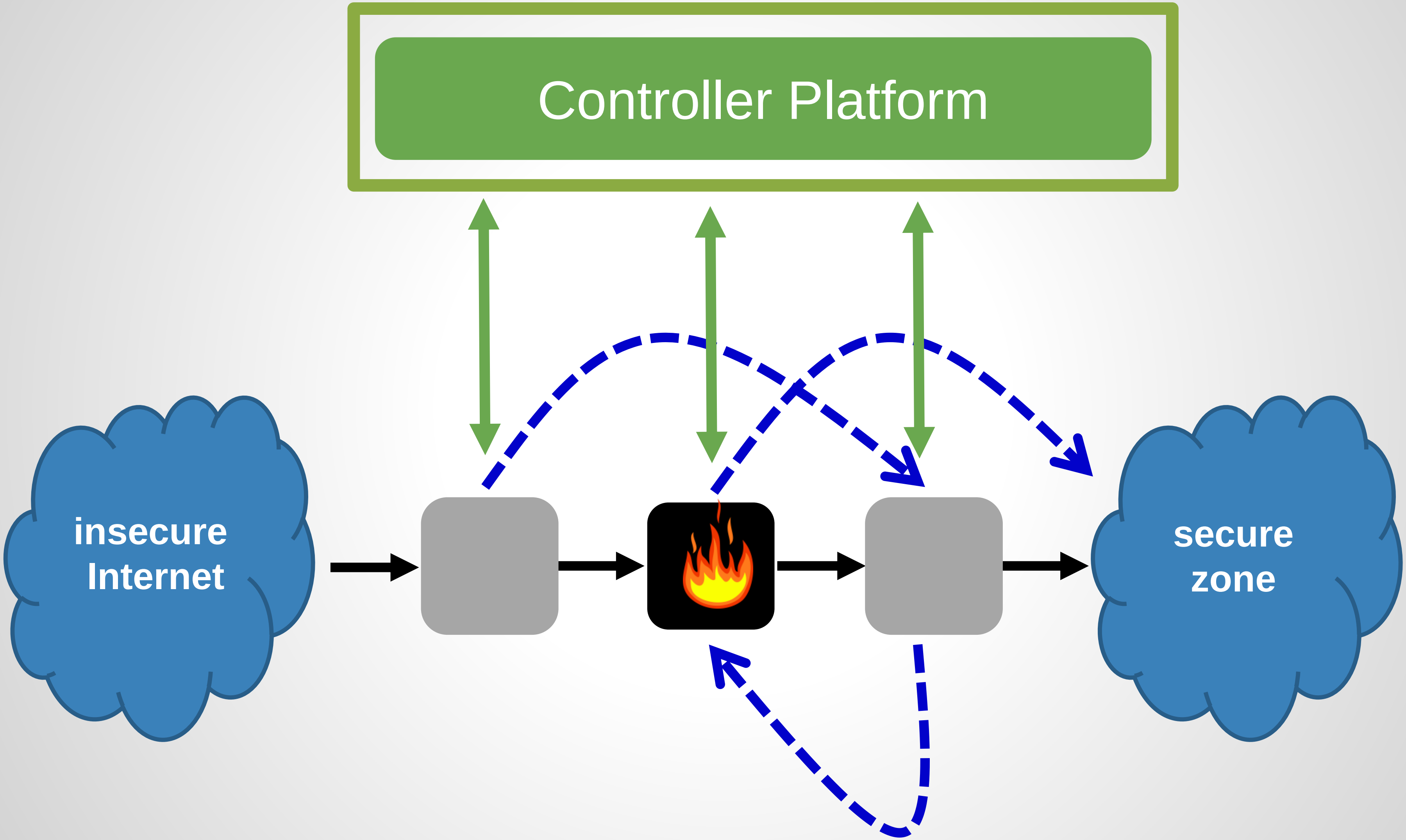




He et al., ACM SOSR 2015:

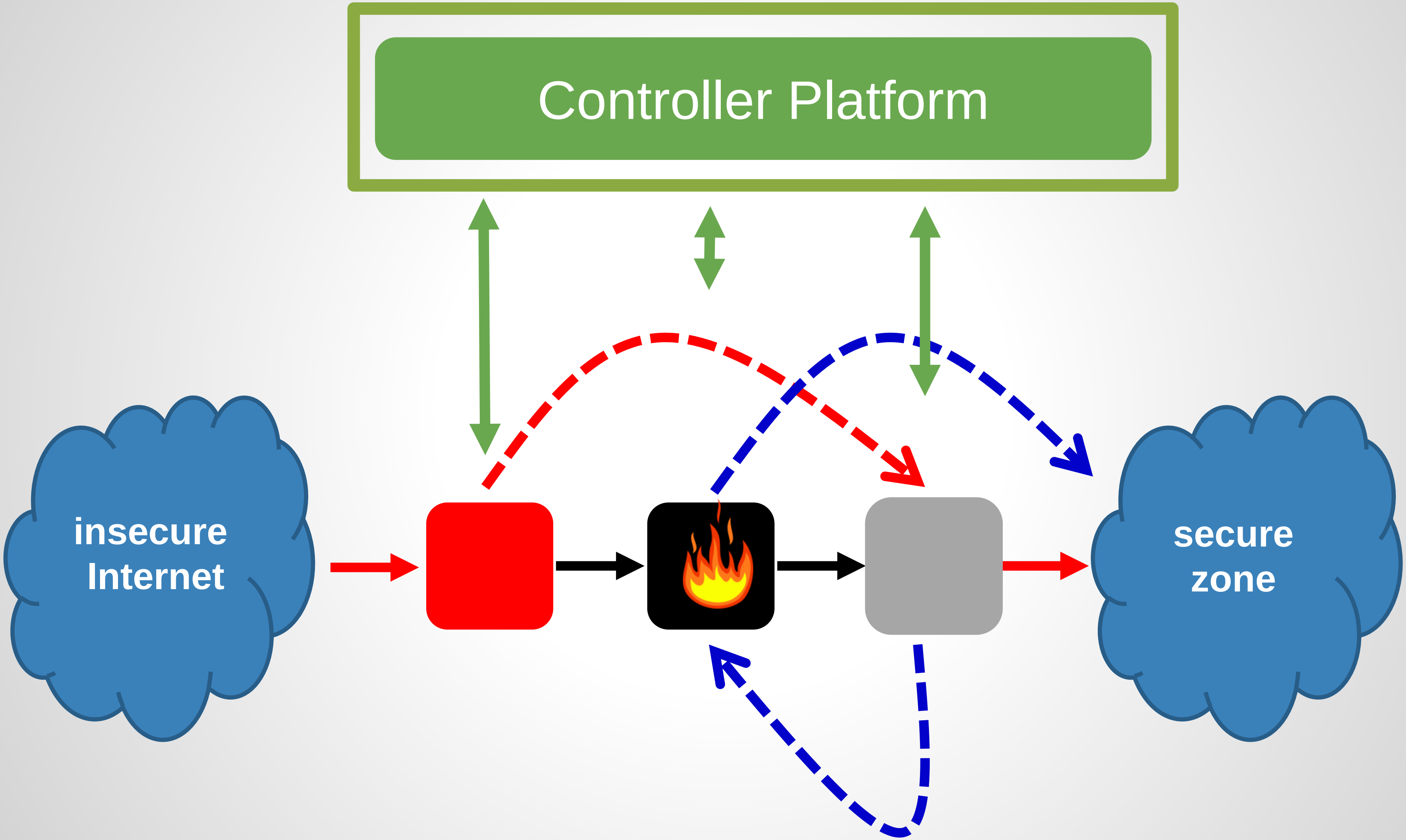
without network latency

What can possibly go wrong?



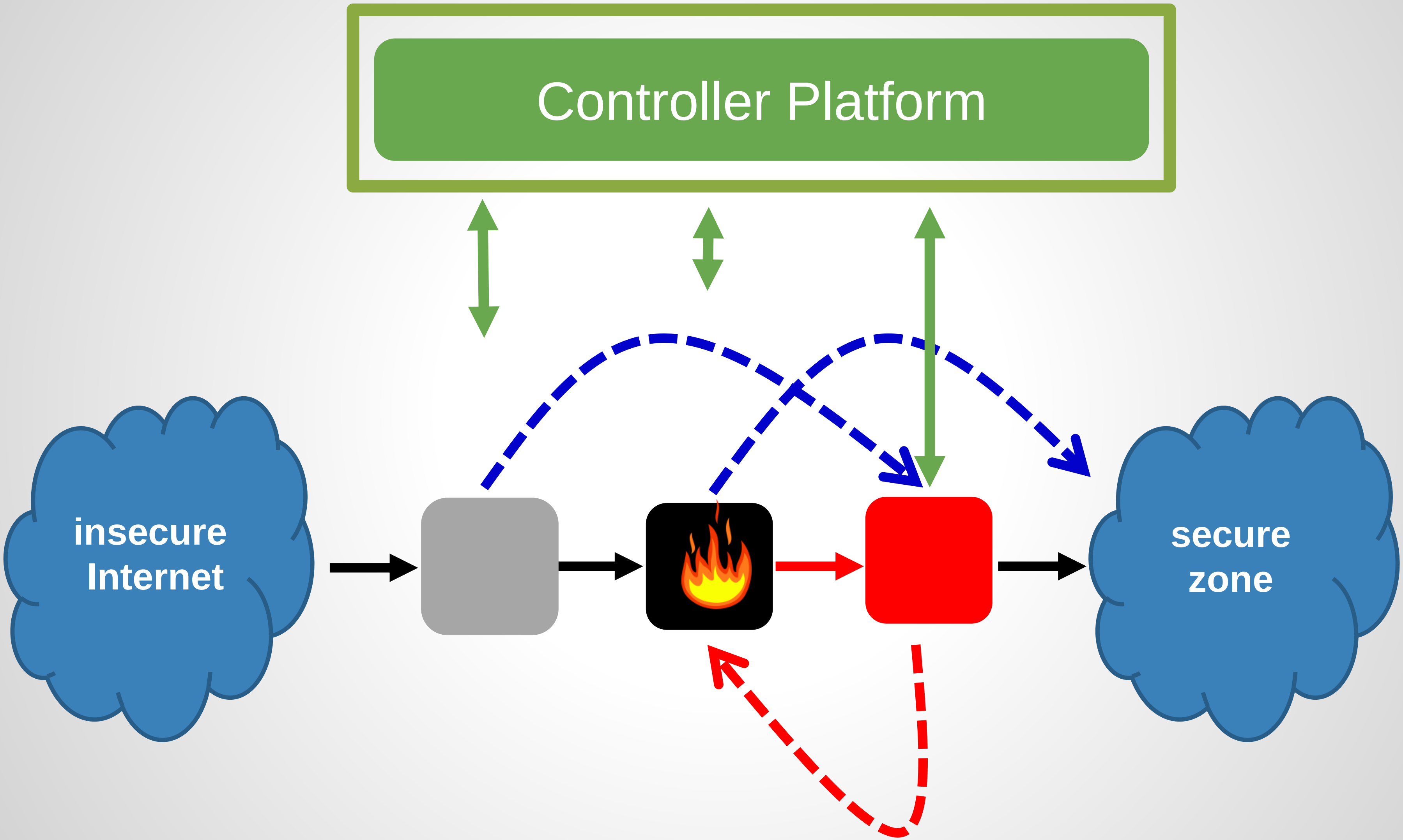
Invariant: Traffic from untrusted hosts to trusted hosts via firewall!

Problem 1: Bypassed Waypoint



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

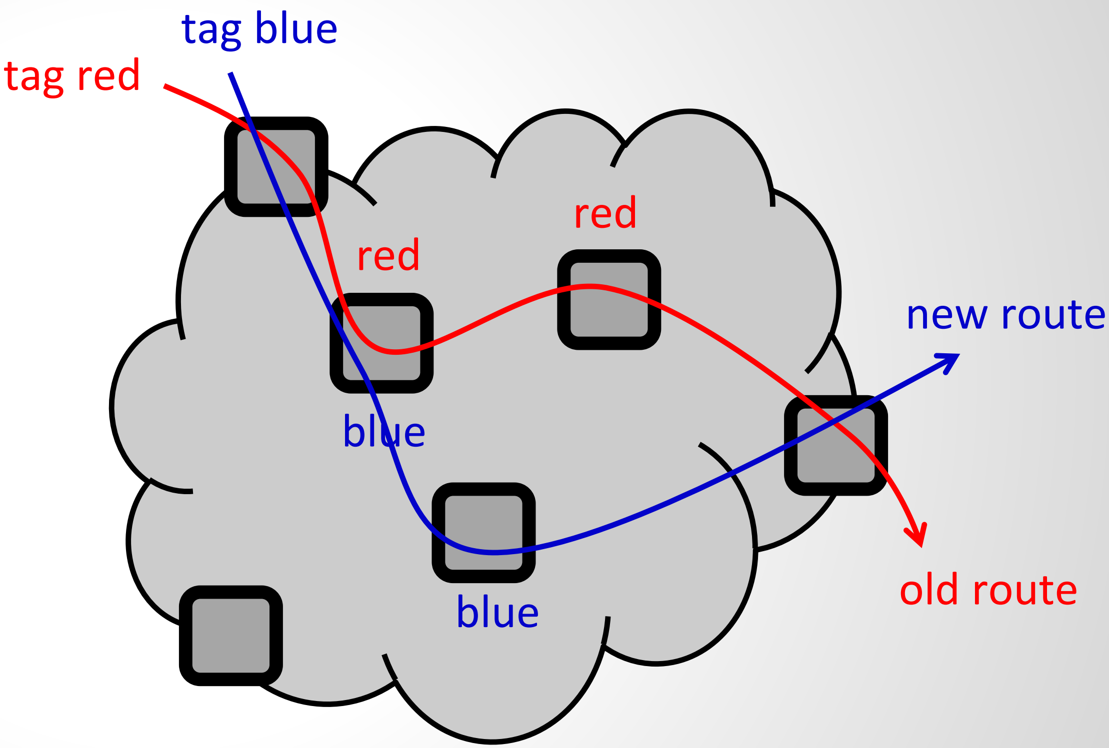
Problem 2: *Transient* Loop



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



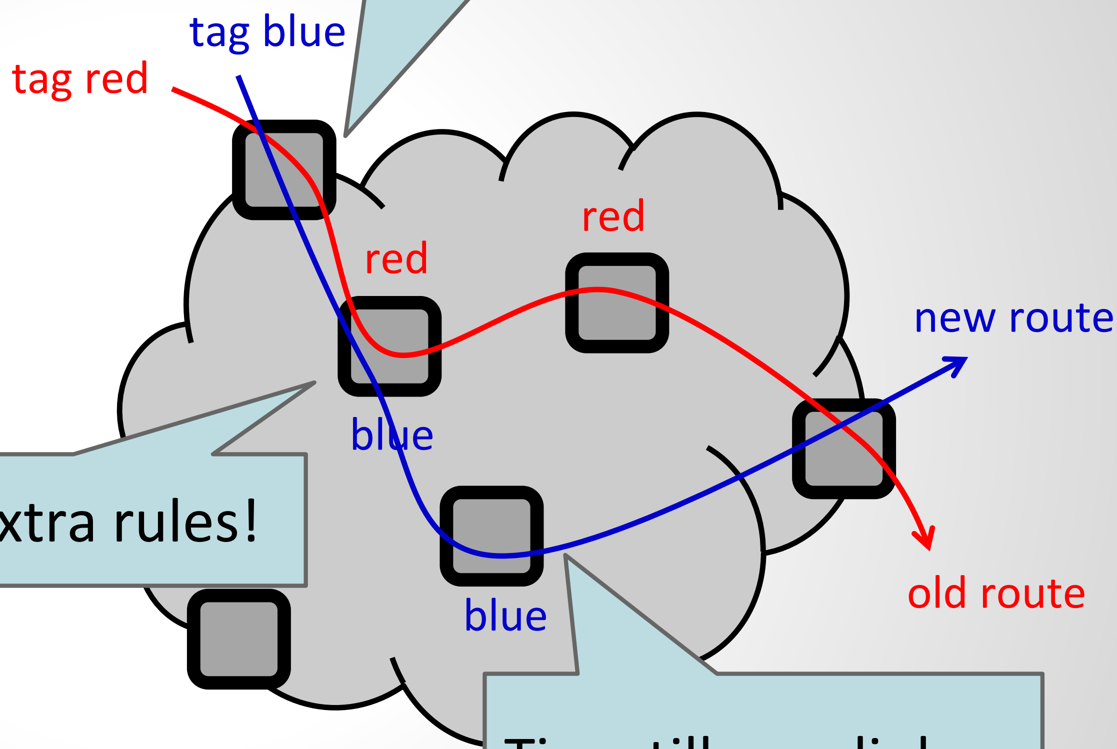
Reitblatt et al. Abstractions for Network Update, ACM SIGCOMM 2012.

Tagging: A Univers

Where to tag?
Header space?
Overhead!

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

Cost of extra rules!



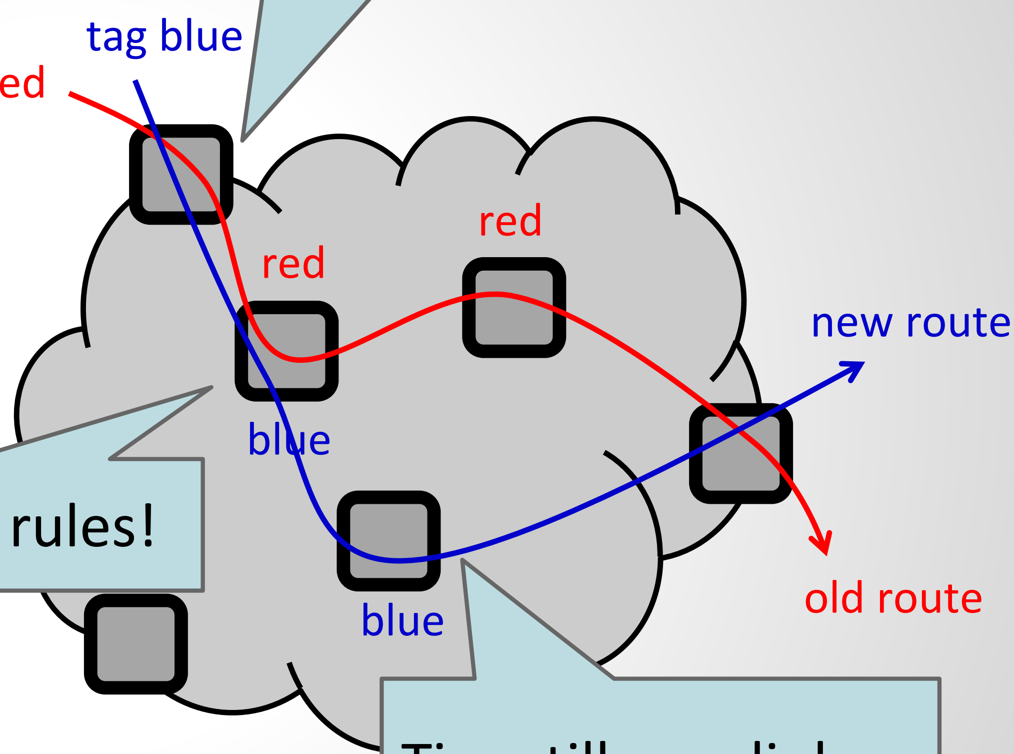
Time till new link becomes available!

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

Tagging: A Univers

- 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!

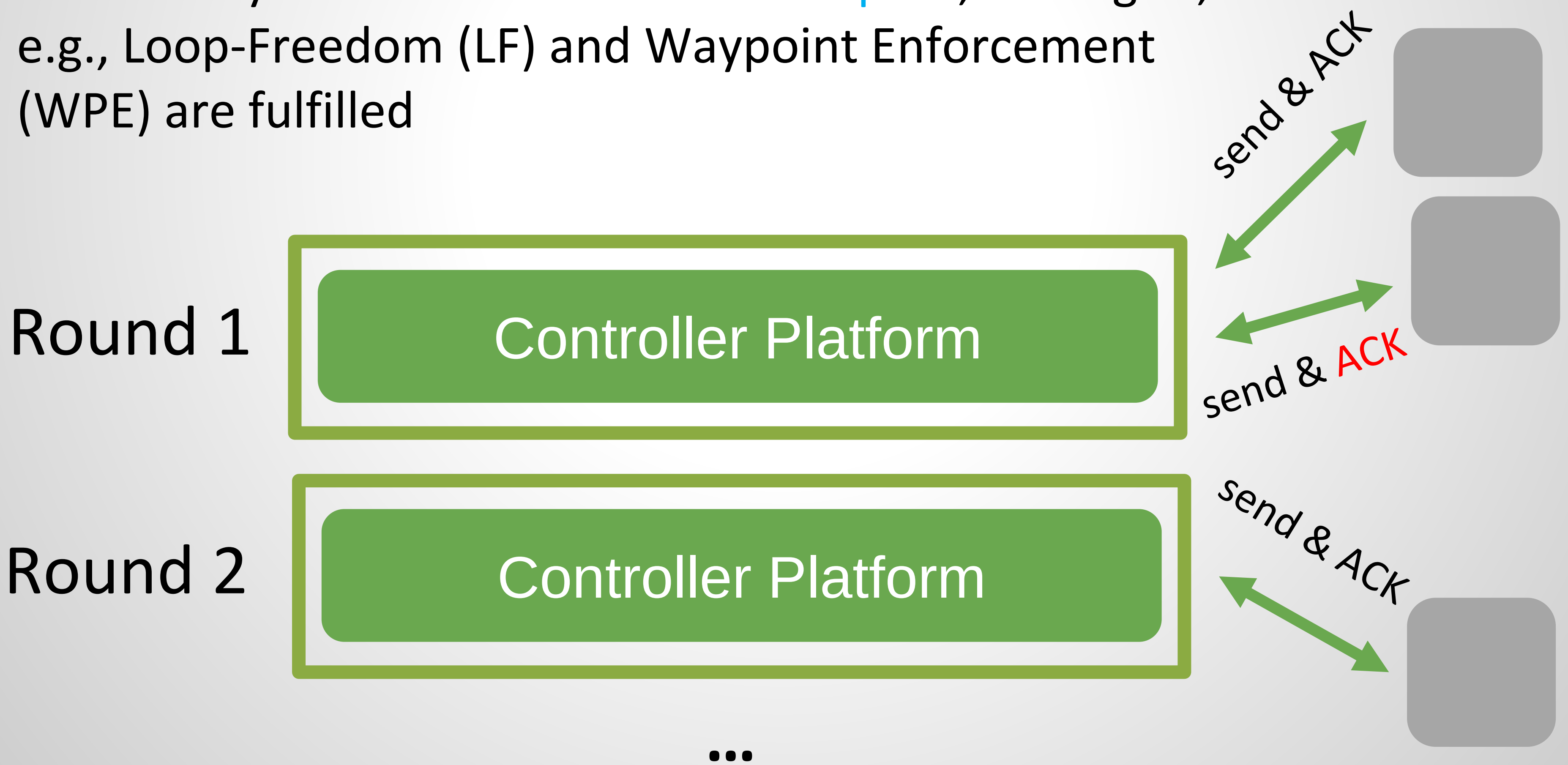
Possible solution without tagging, and at least preserve weaker consistency properties?

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

Idea: Schedule Subsets of Nodes!

Idea: Schedule safe update subsets in **multiple rounds!**

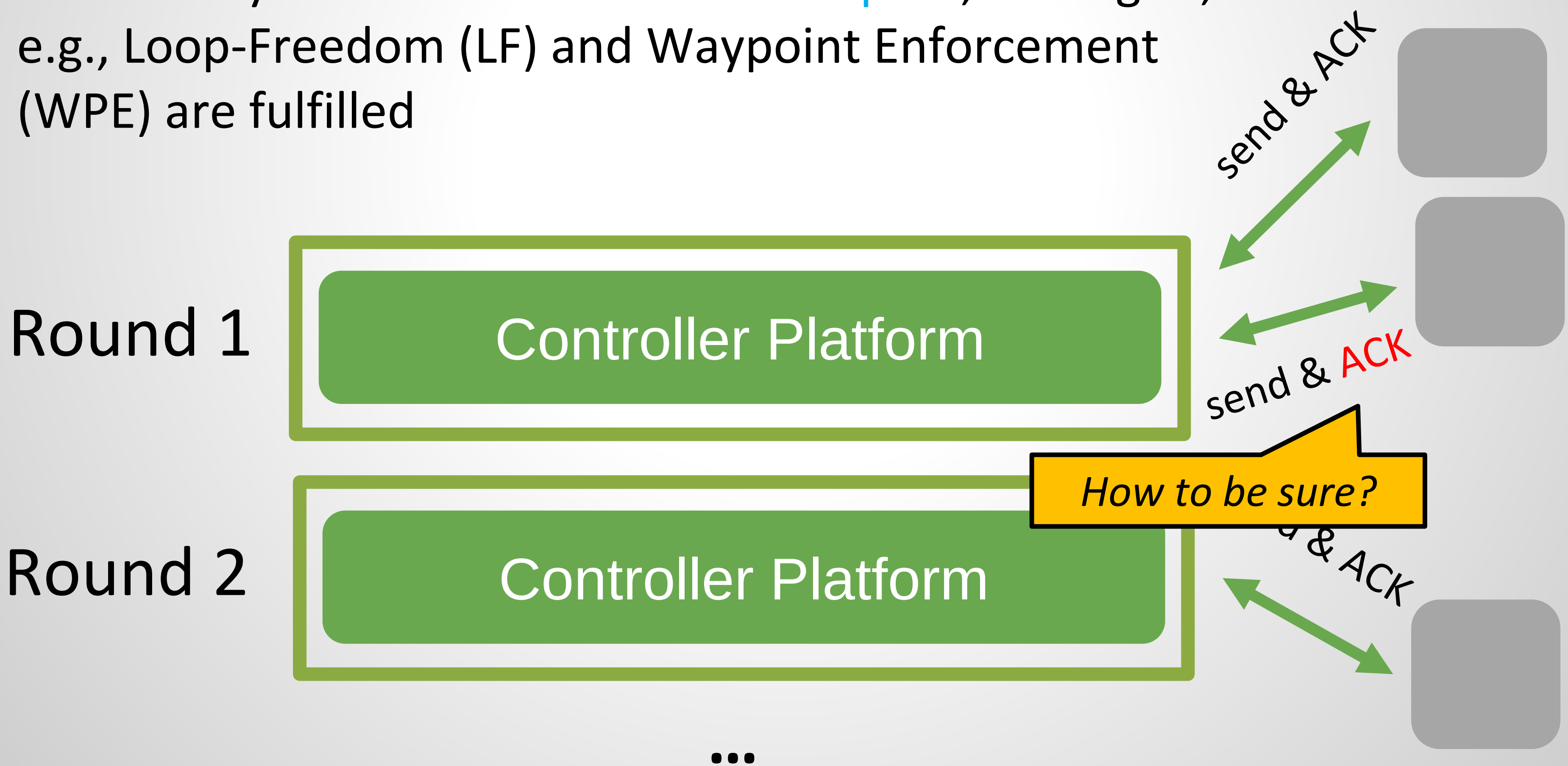
Packet may take a mix of old and new path, as long as, e.g., Loop-Freedom (LF) and Waypoint Enforcement (WPE) are fulfilled



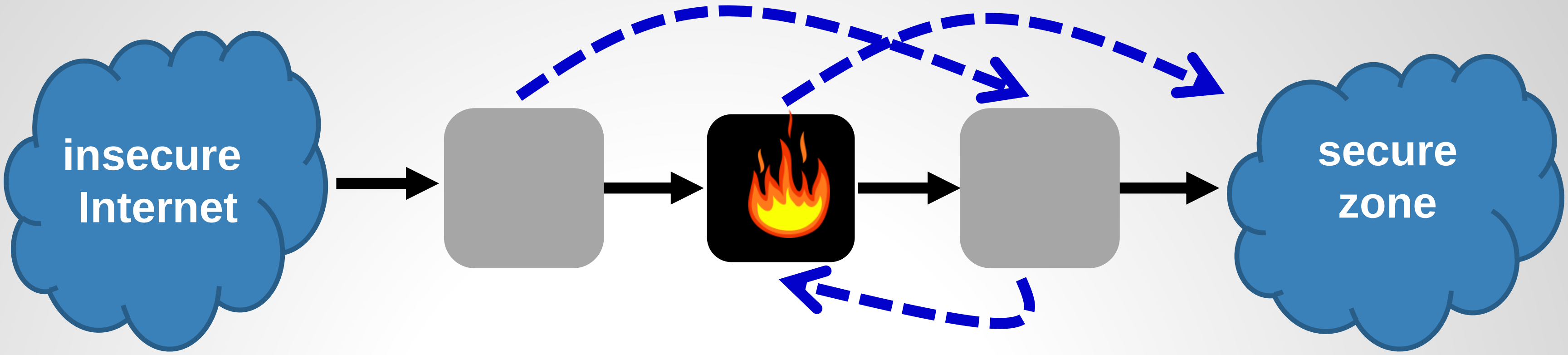
Idea: Schedule Subsets of Nodes!

Idea: Schedule safe update subsets in **multiple rounds!**

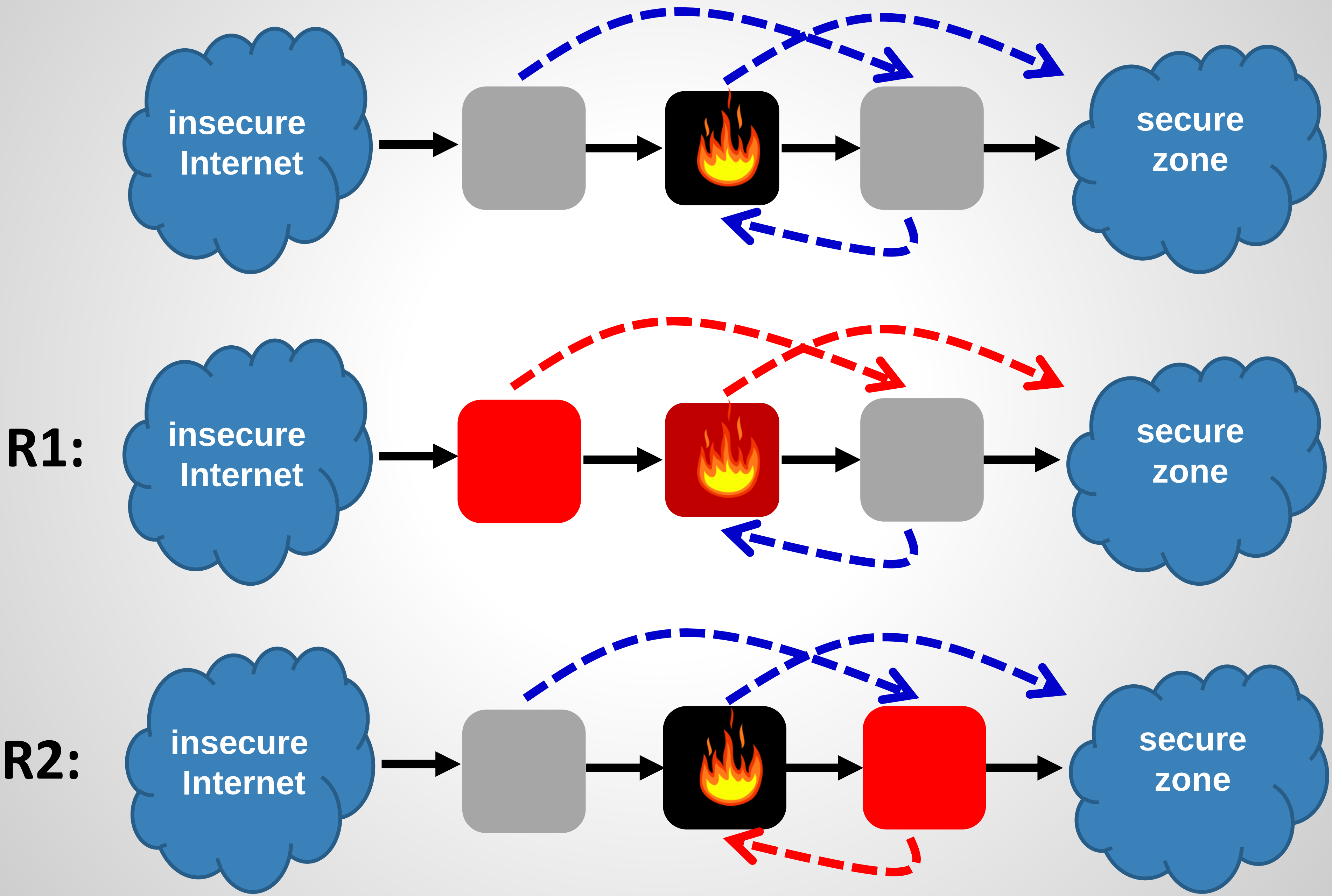
Packet may take a mix of old and new path, as long as, e.g., Loop-Freedom (LF) and Waypoint Enforcement (WPE) are fulfilled



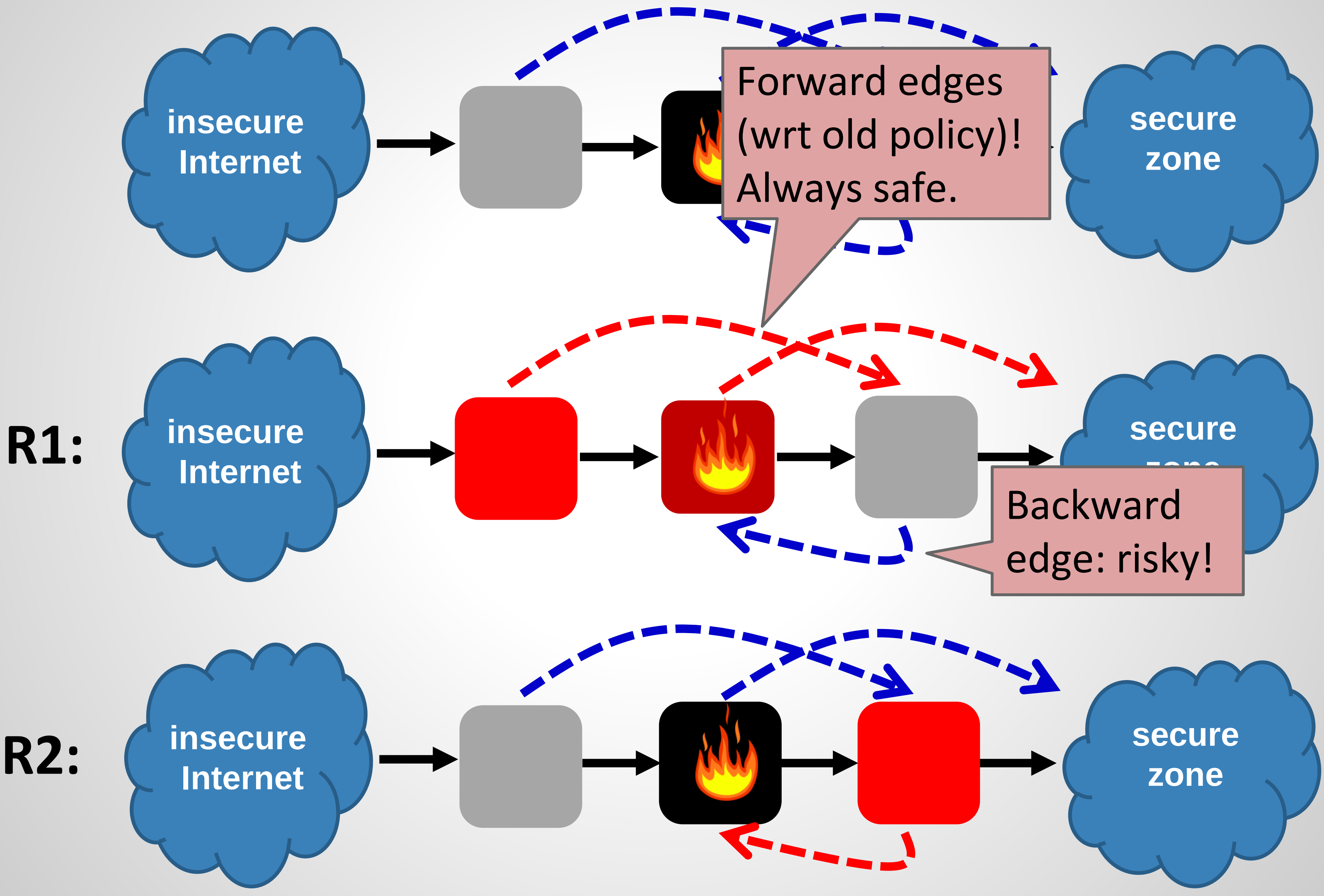
Going Back to Our Examples: LF Update



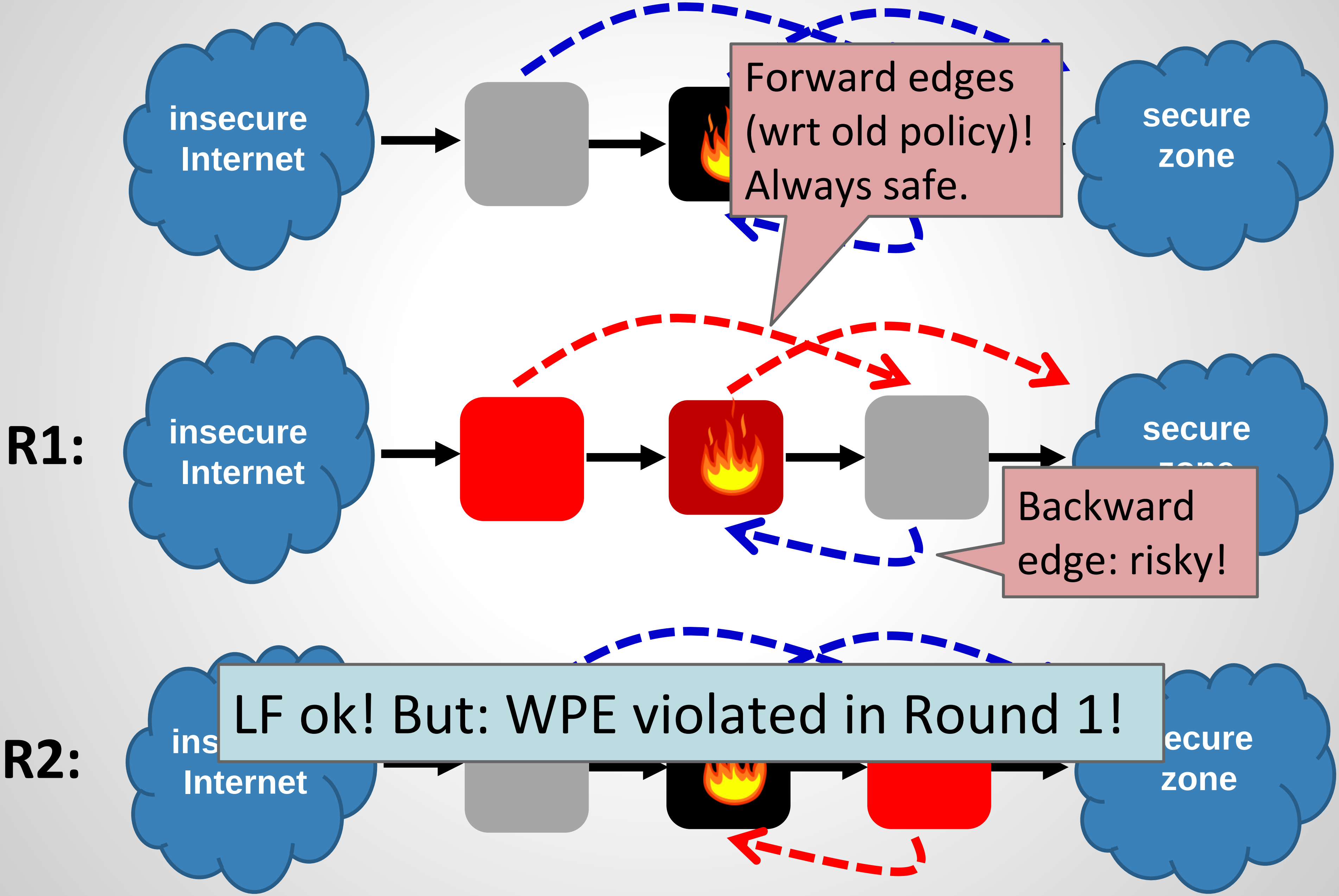
Going Back to Our Examples: LF Update



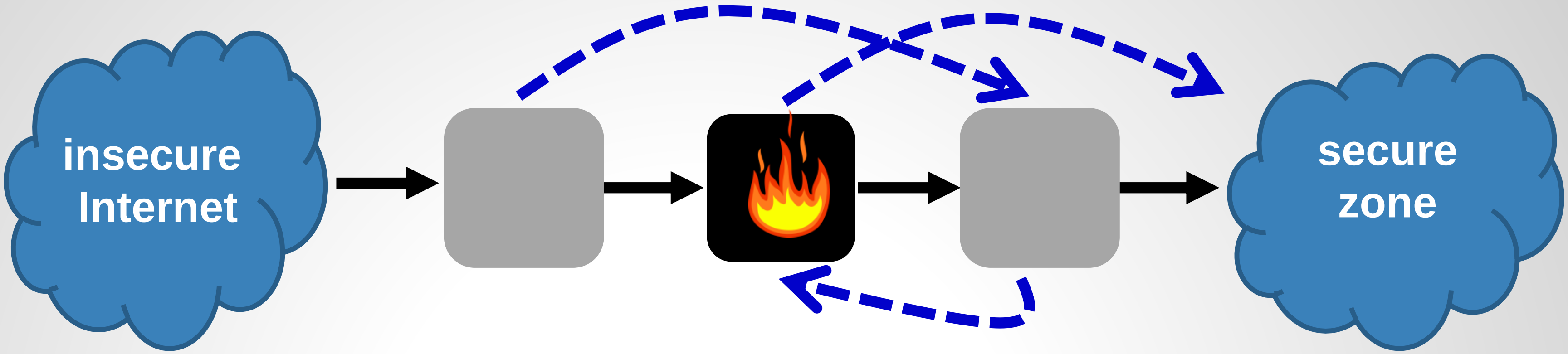
Going Back to Our Examples: LF Update



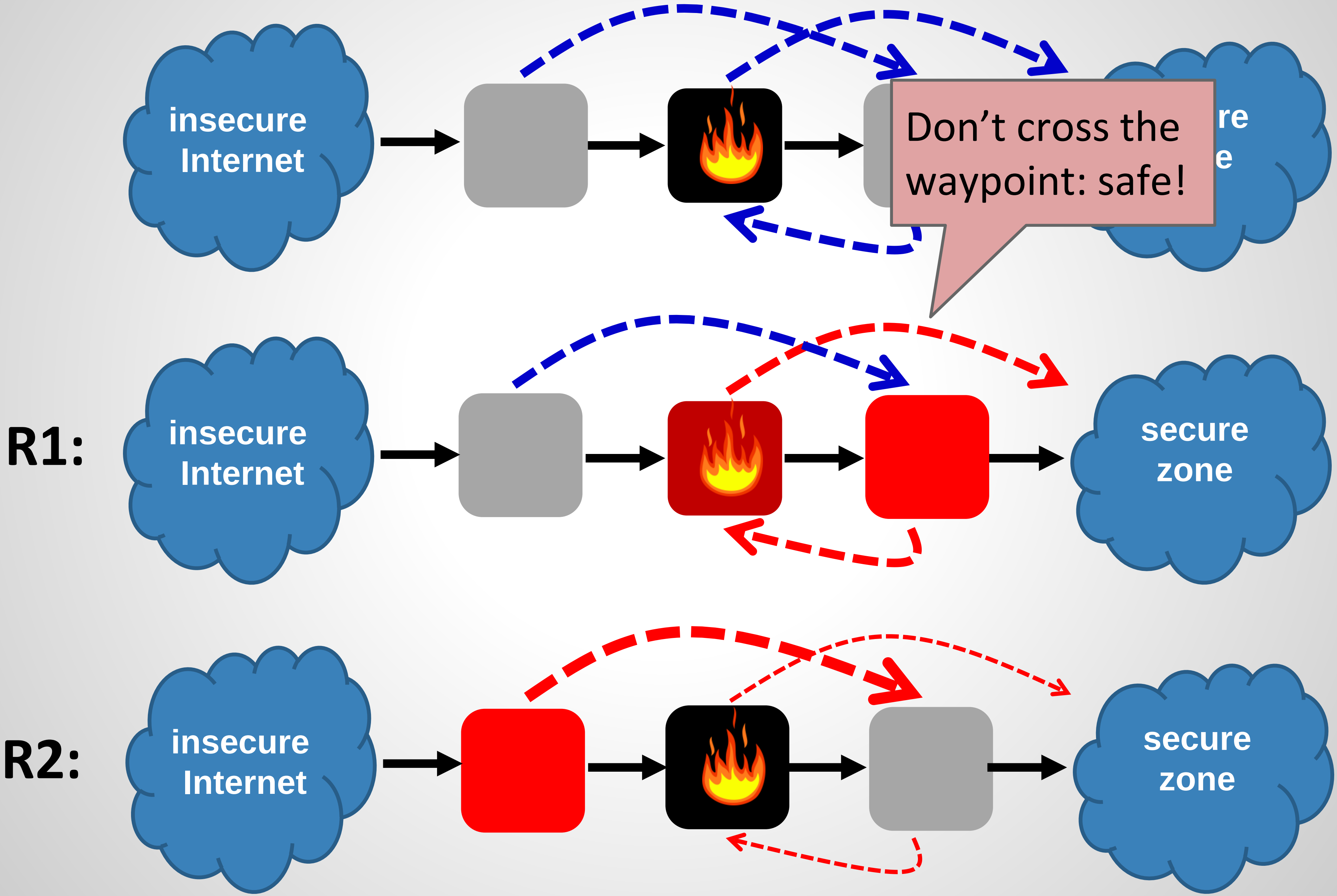
Going Back to Our Examples: LF Update



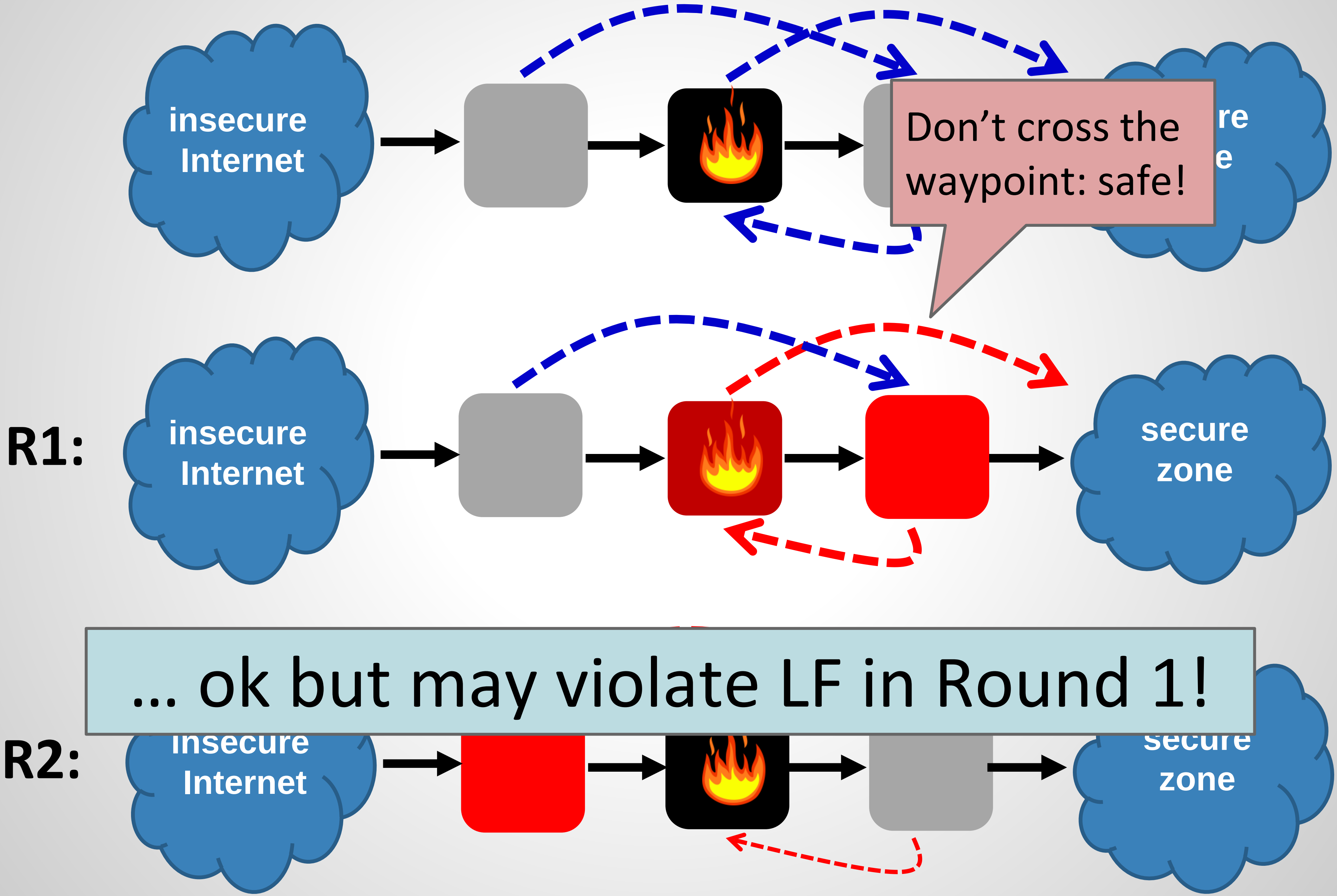
Going Back to Our Examples: WPE Update



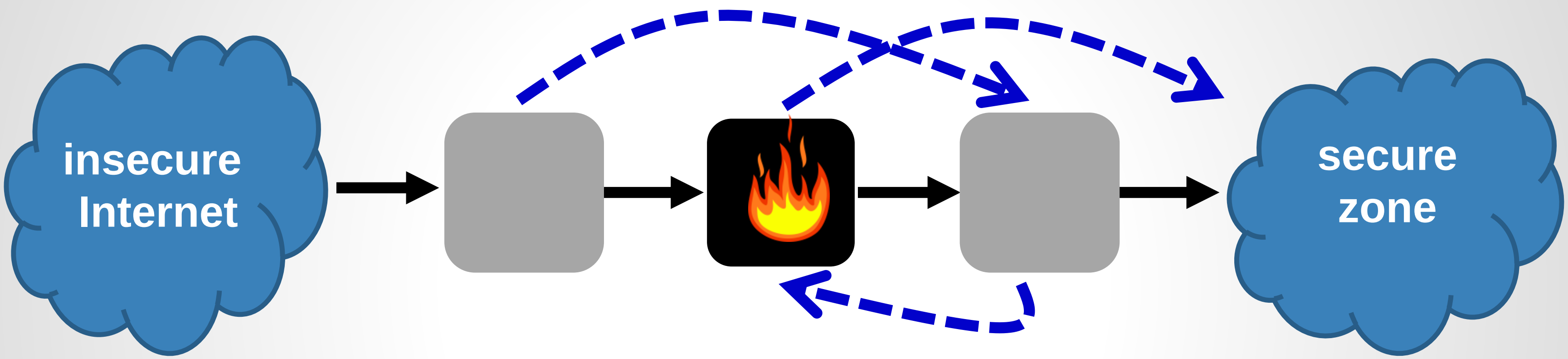
Going Back to Our Examples: WPE Update



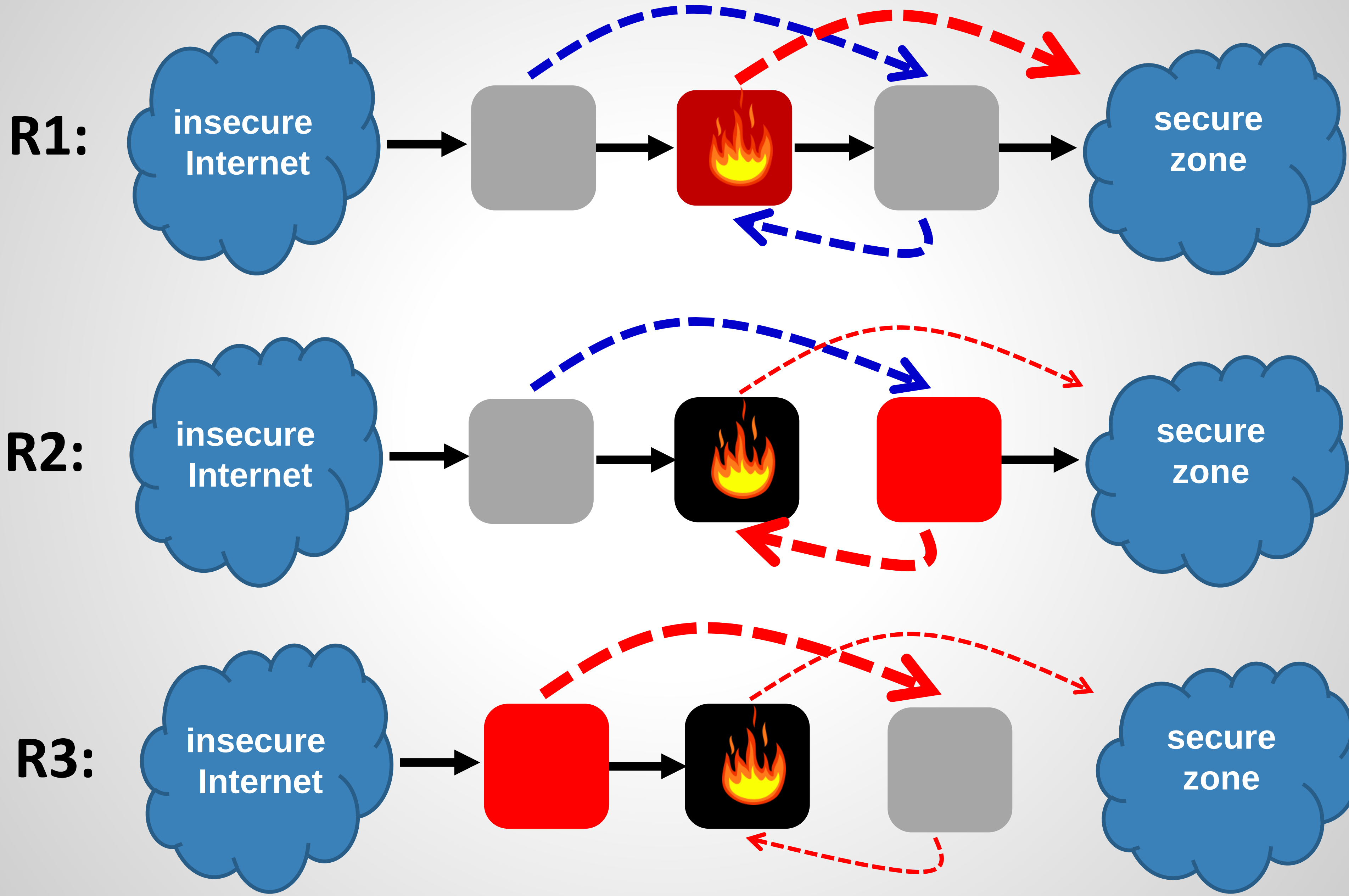
Going Back to Our Examples: WPE Update



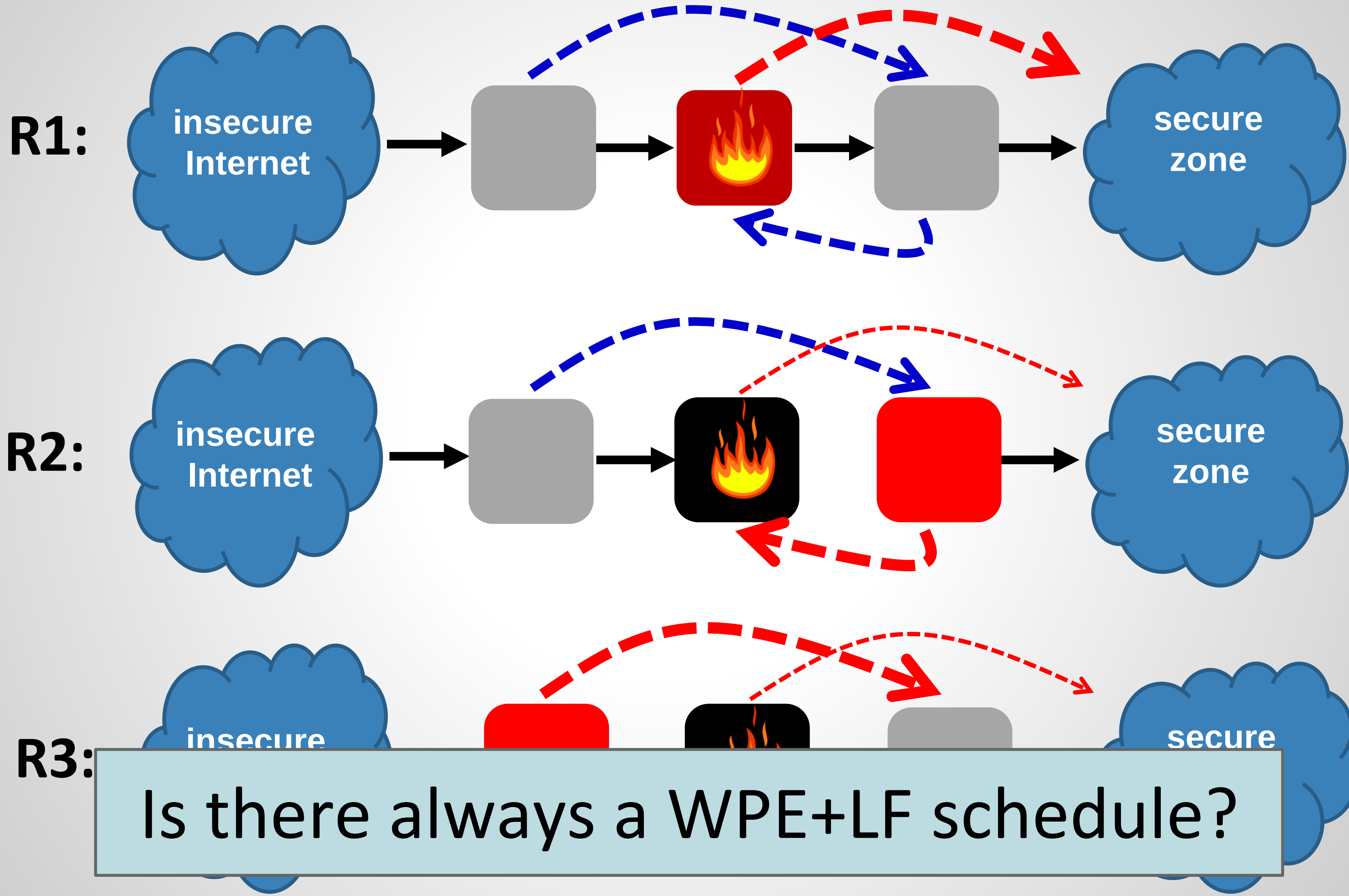
Going Back to Our Examples: Both WPE+LF?



Going Back to Our Examples: WPE+LF!

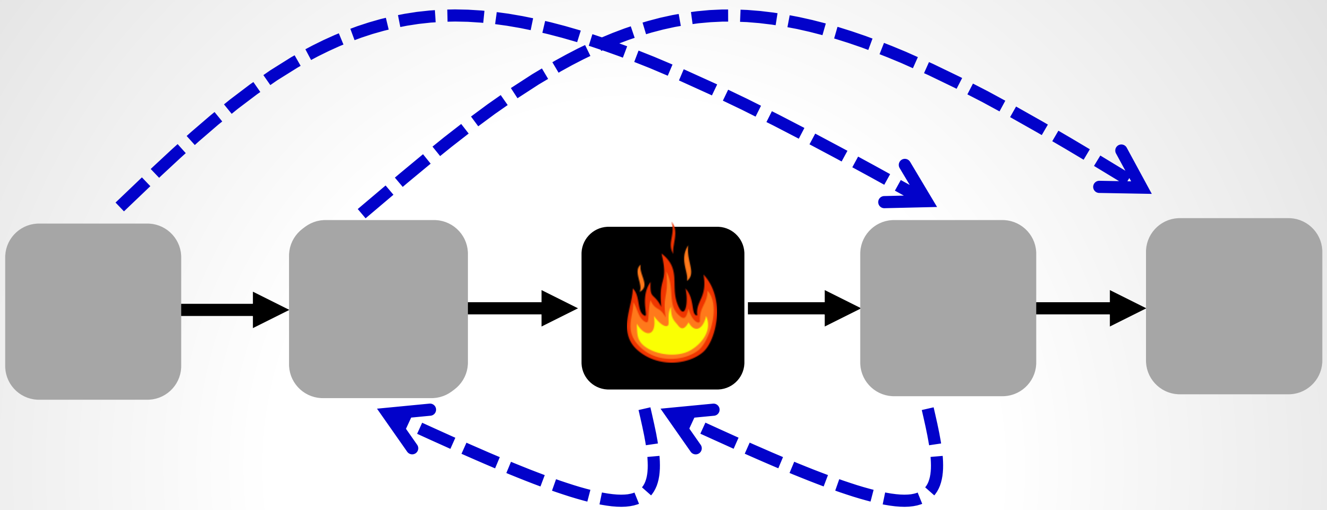


Going Back to Our Examples: WPE+LF!

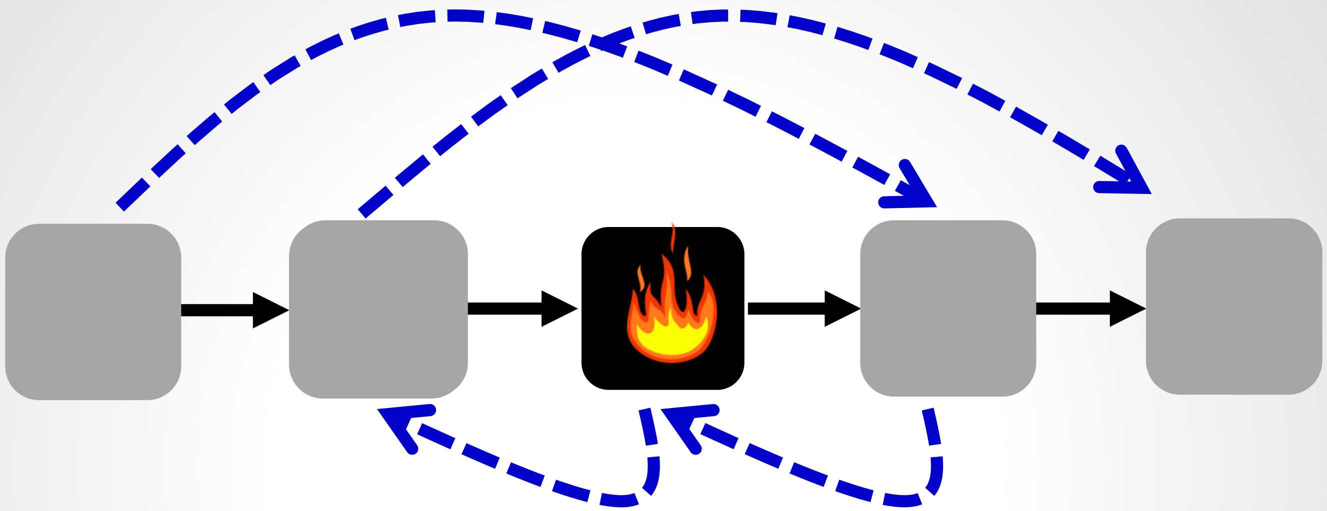


Is there always a WPE+LF schedule?

What about this one?



LF and WPE may conflict!

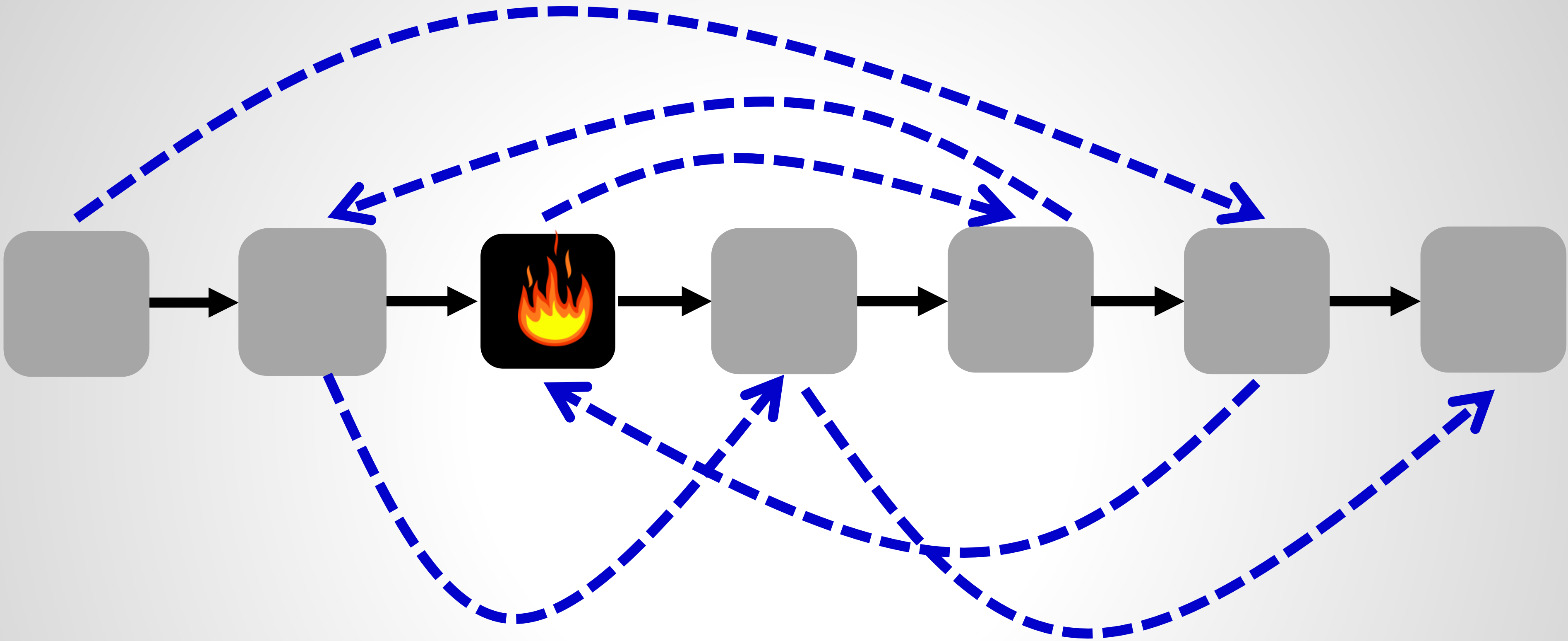


- ❑ Cannot update any **forward edge** in R1: WP
- ❑ Cannot update any **backward edge** in R1: LF

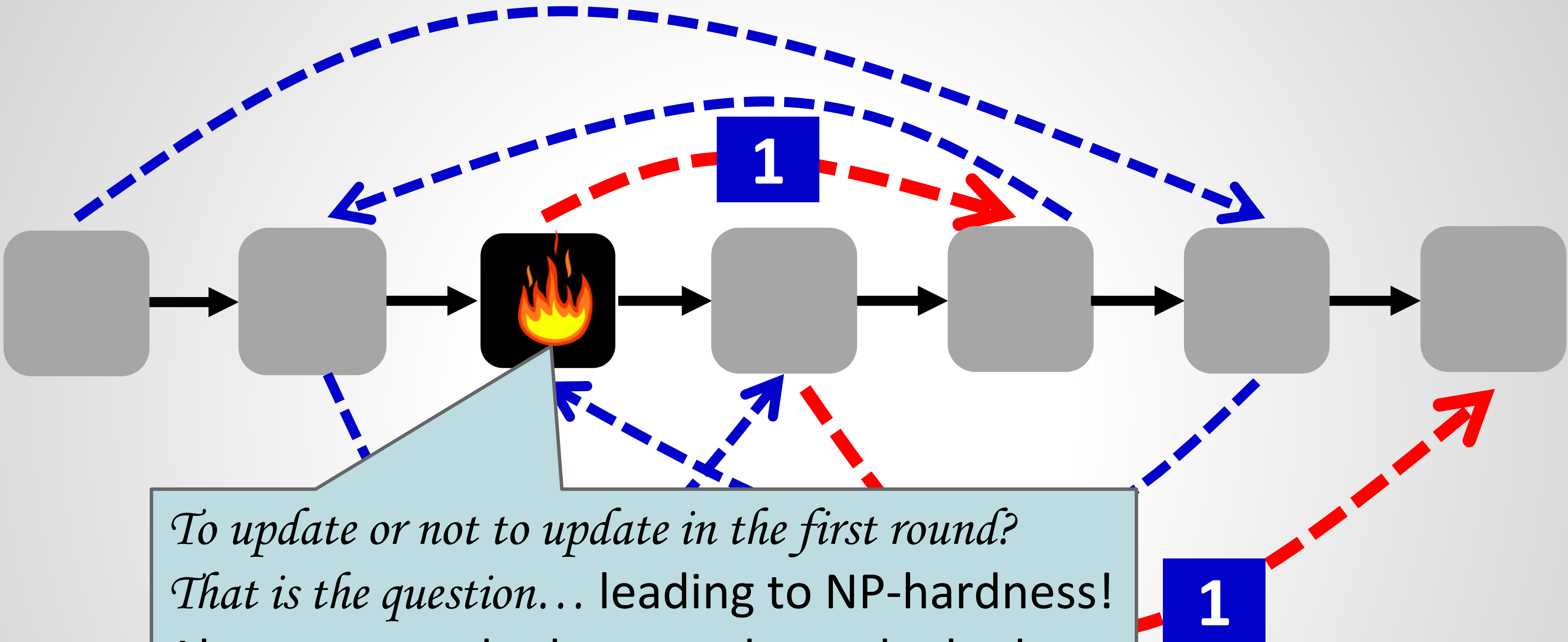
No schedule exists!

Resort to tagging…

What about this one?



NP-Hard!



To update or not to update in the first round?
That is the question… leading to NP-hardness!
Also an example that greedy can be bad.

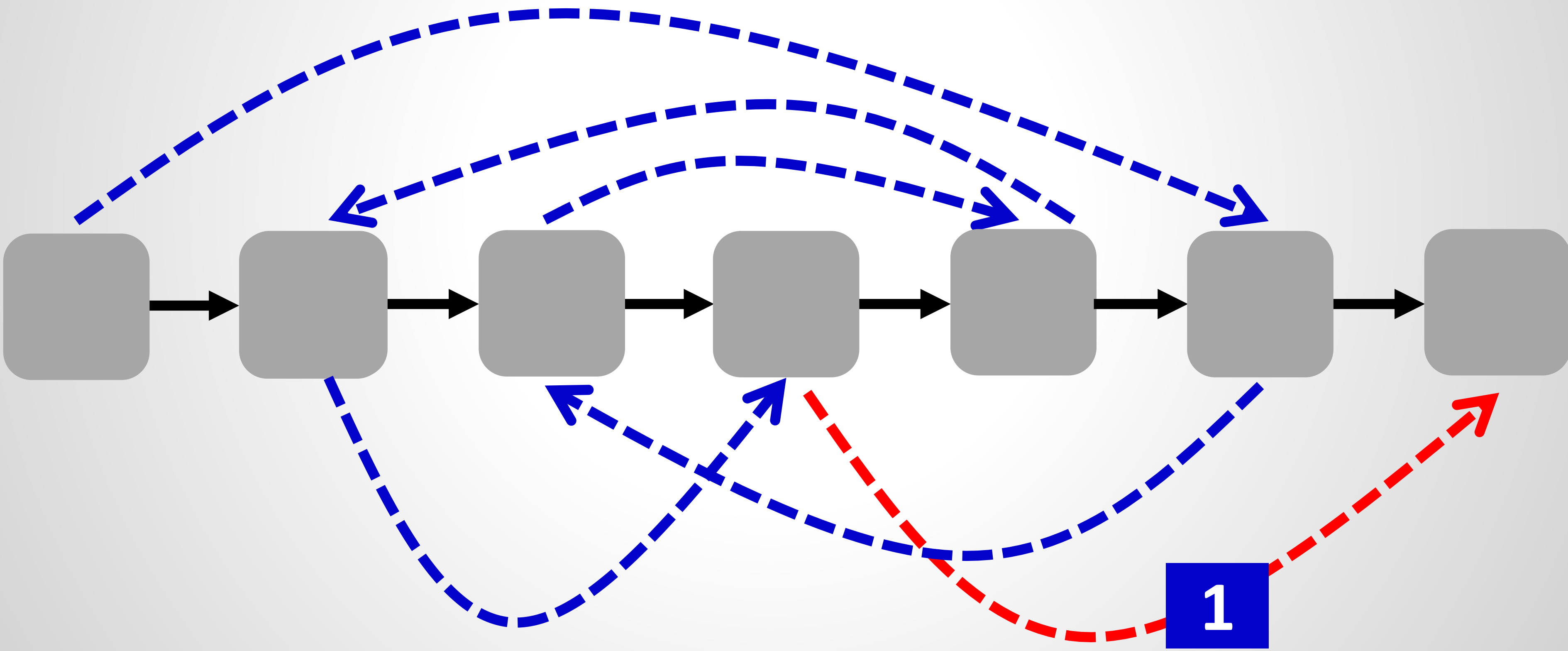
Bad news: Even decidability hard: cannot quickly test feasibility and if infeasible resort to say, tagging solution!

We don't know!

Open question: very artificial? Under which circumstances poly-time?

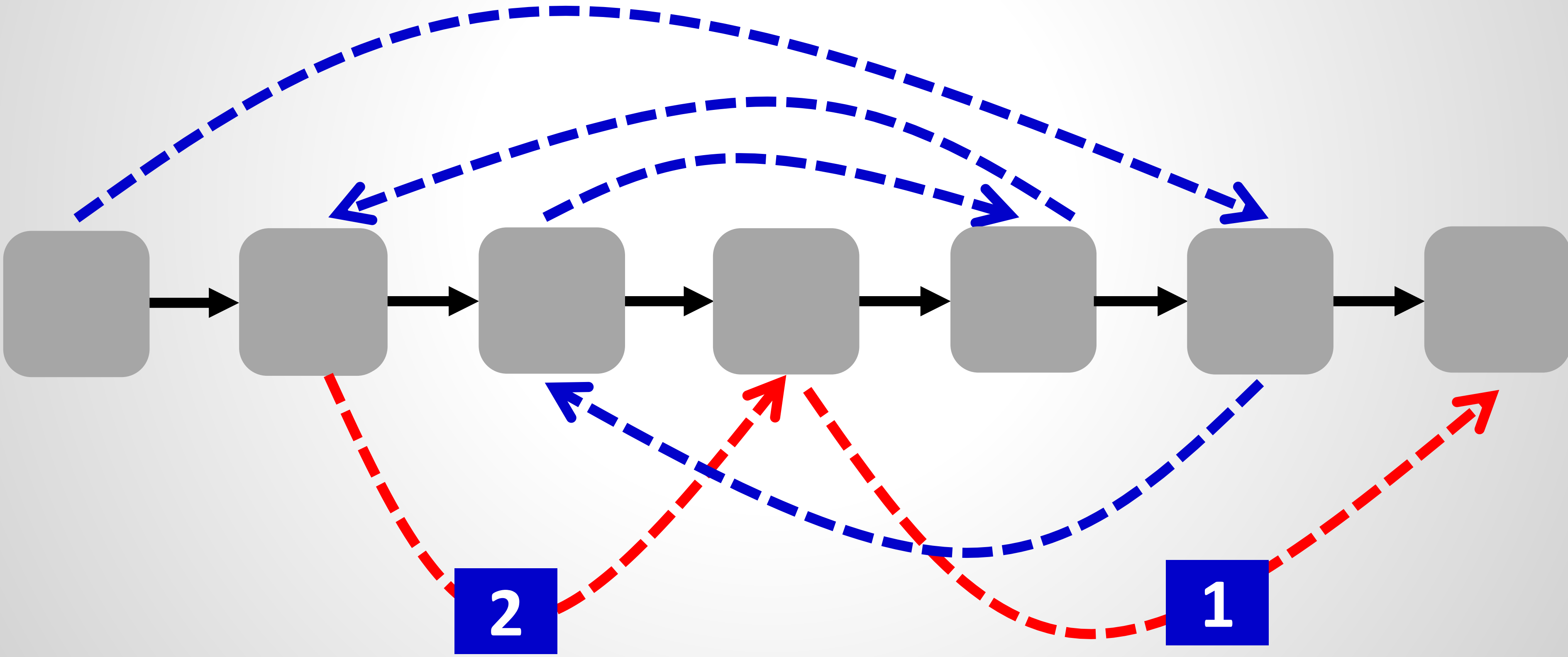
Let us focus on loop-freedom only: always possible in *n* rounds! How?

Let us focus on loop-freedom only: always possible in *n* rounds! How?



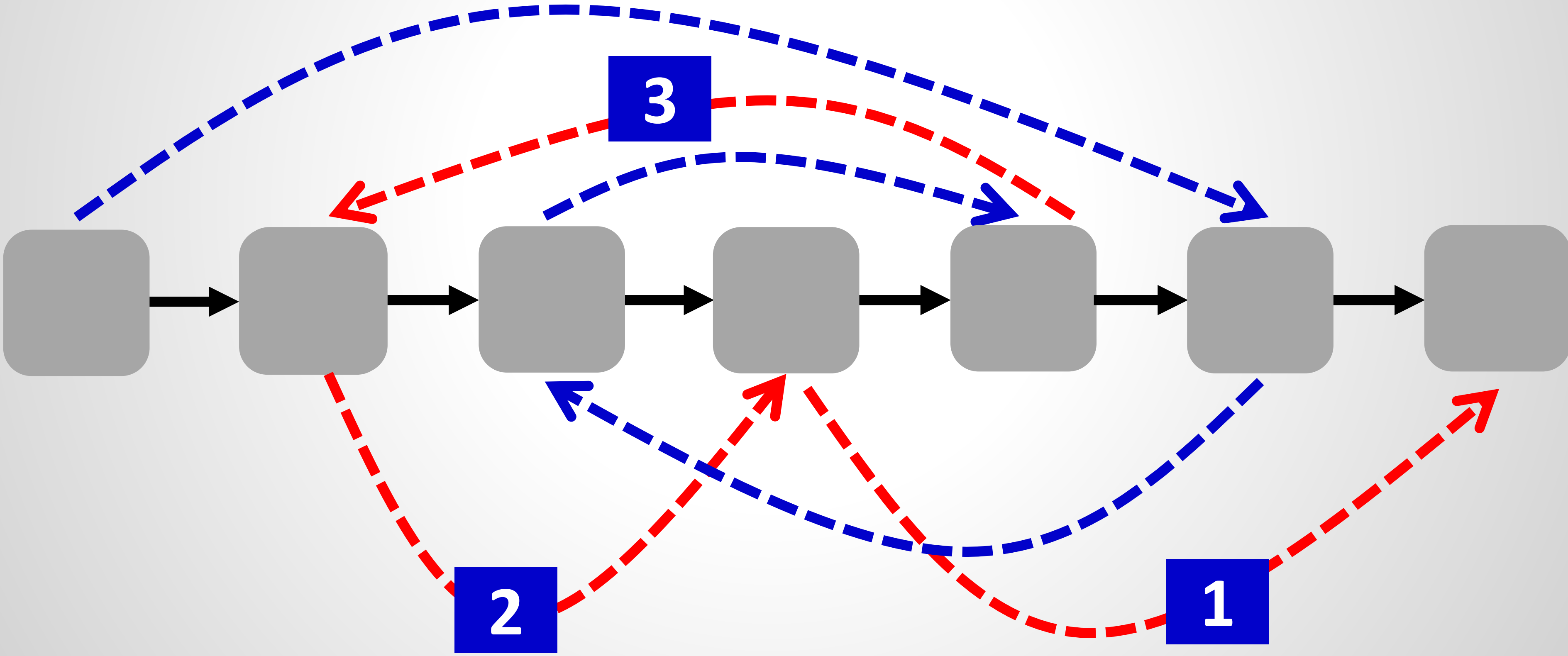
From the destination! Invariant: path suffix updated!

Let us focus on loop-freedom only: always possible in *n* rounds! How?



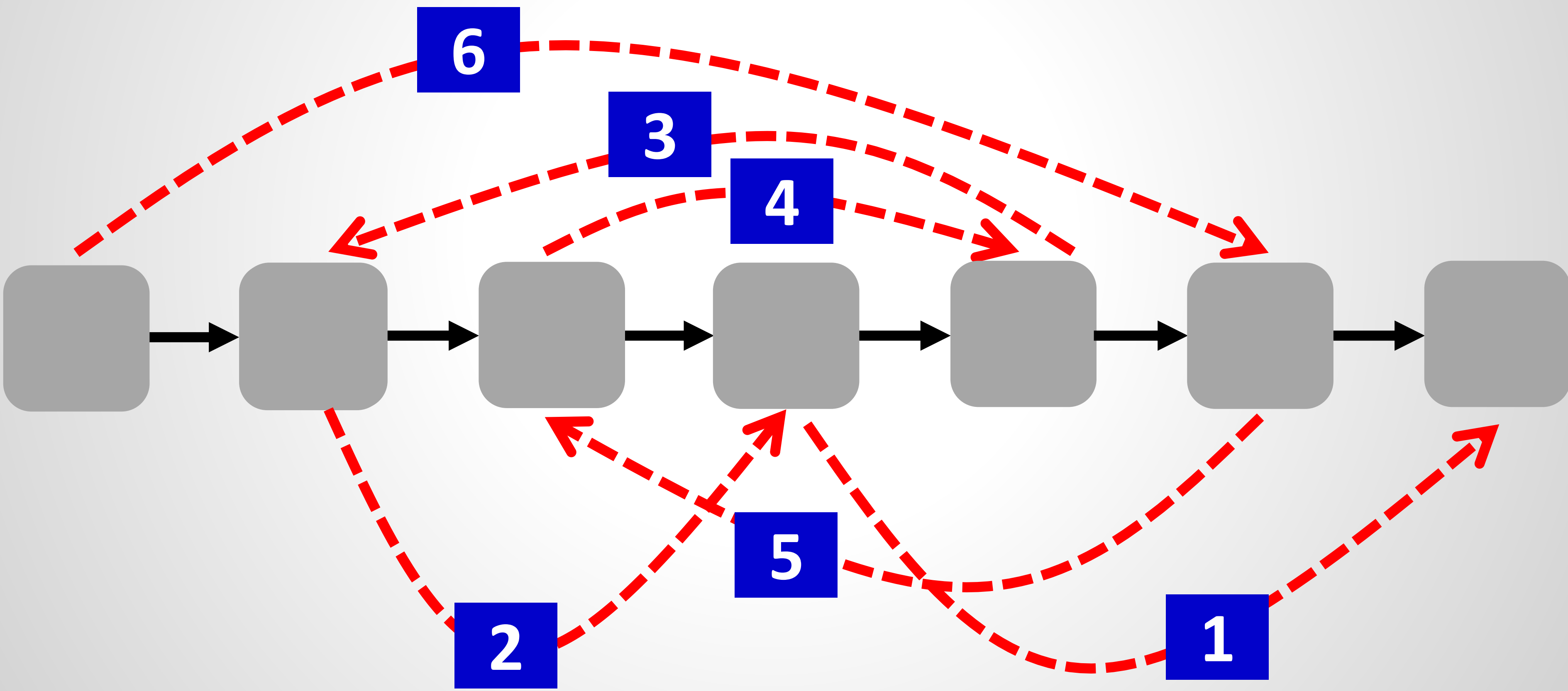
From the destination! Invariant: path suffix updated!

Let us focus on loop-freedom only: always possible in *n* rounds! How?



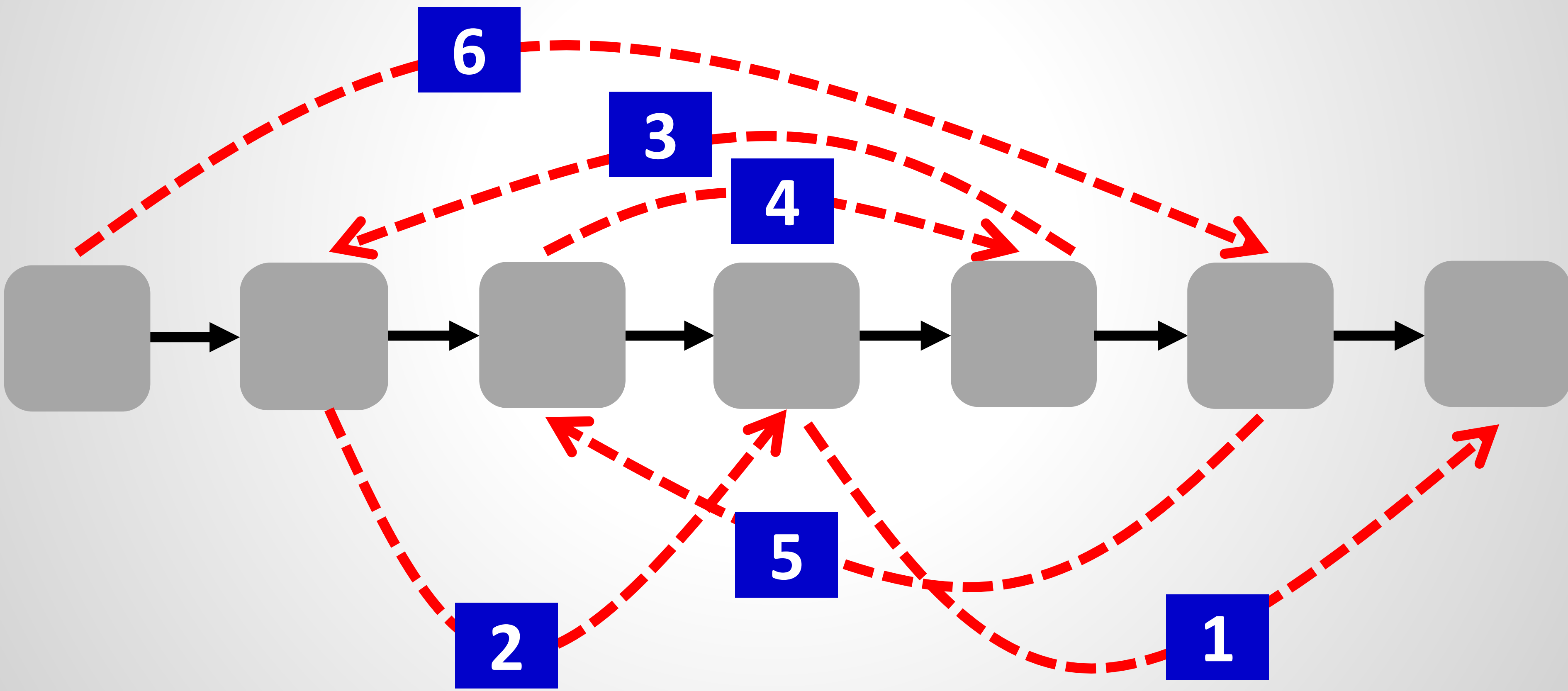
From the destination! Invariant: path suffix updated!

Let us focus on loop-freedom only:
always possible in *n* rounds! How?



From the destination! Invariant: path suffix updated!

Let us focus on loop-freedom only: always possible in *n* rounds! How?



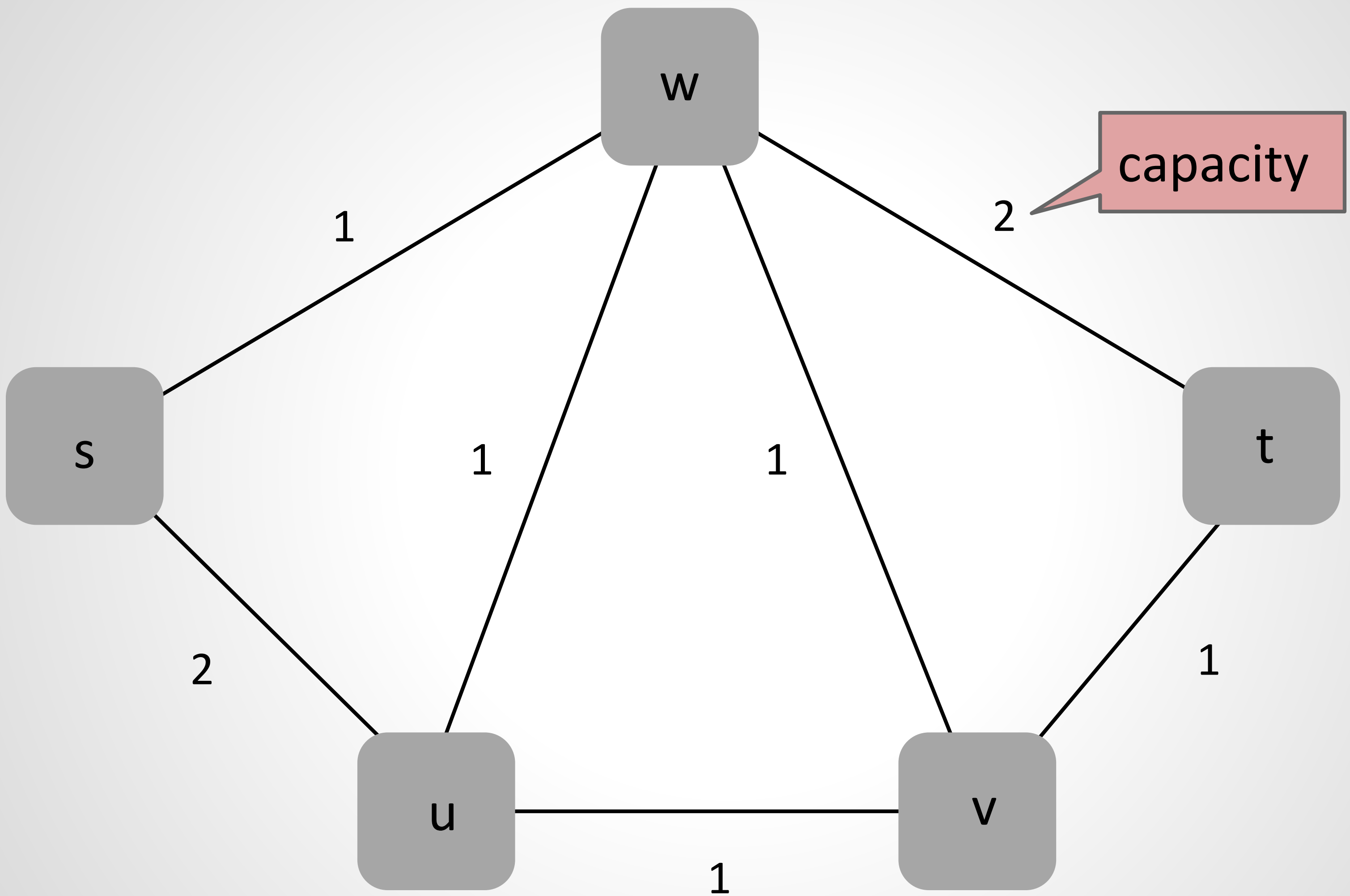
From the destination! Invariant: path suffix updated!

But how to minimize # rounds?

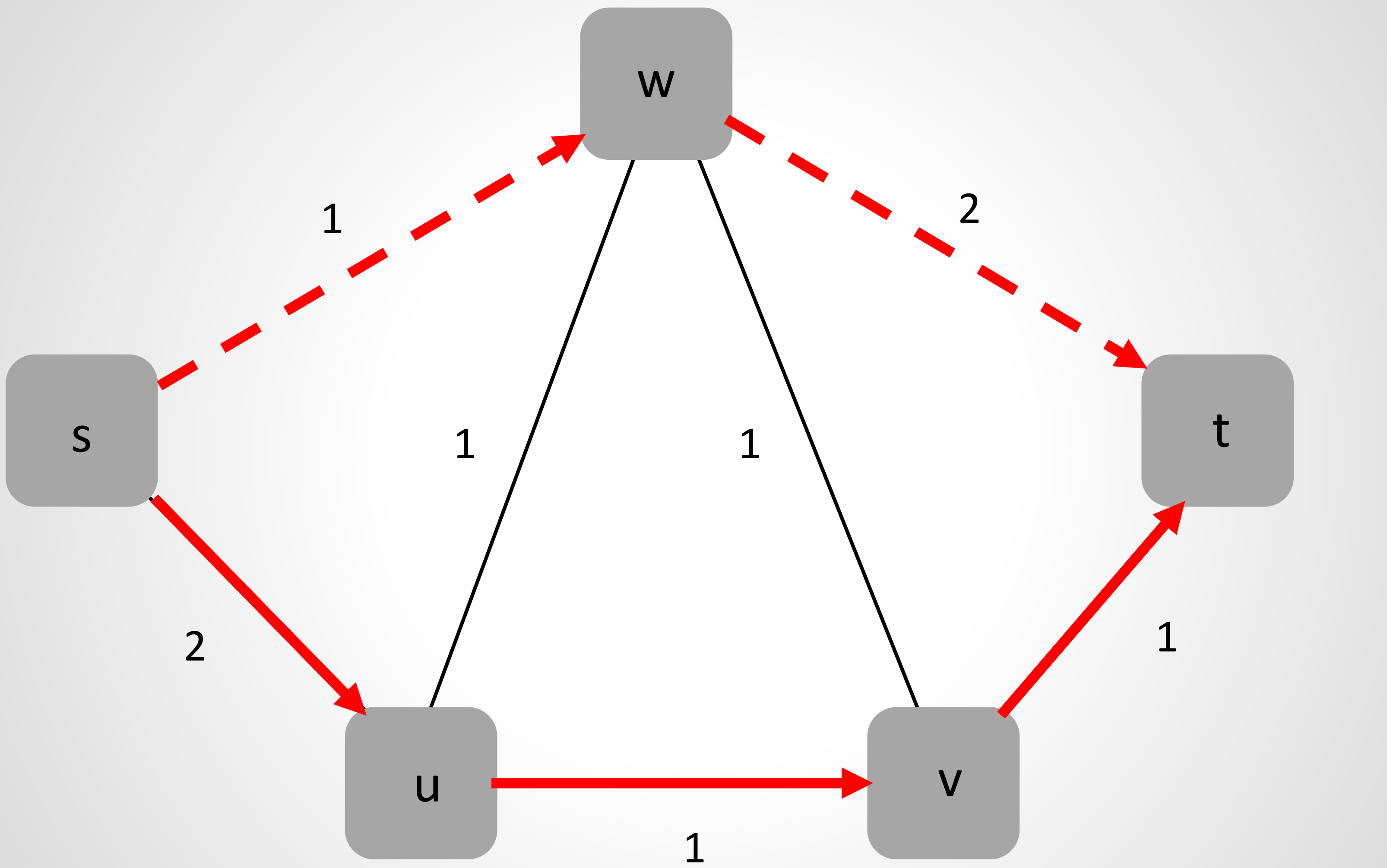
But how to minimize # rounds?

2 rounds easy, 3 rounds NP-hard. Let’s take it offline!

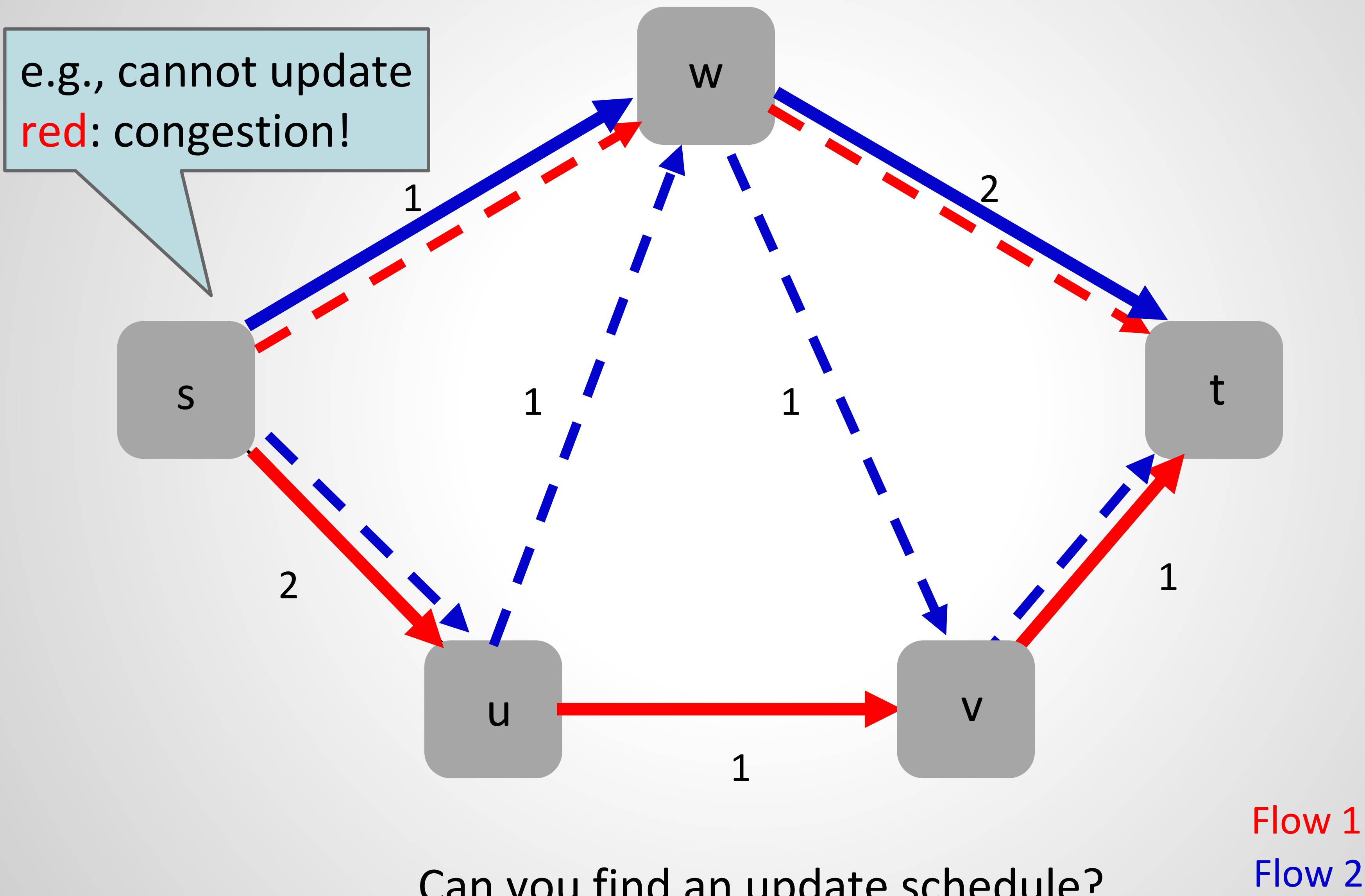
What about capacity constraints?



What about capacity constraints?

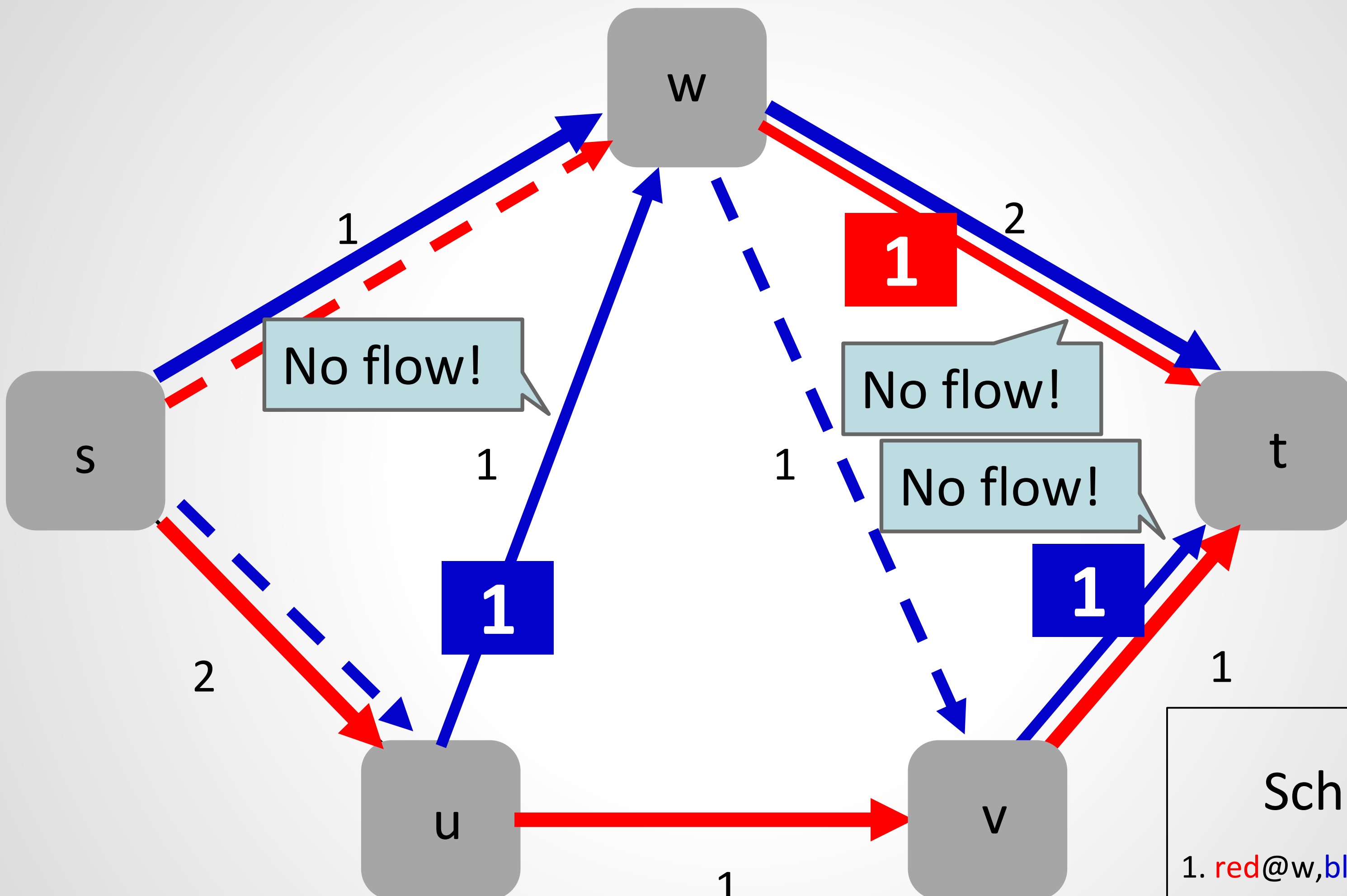


What about capacity constraints?



Can you find an update schedule?

What about capacity constraints?

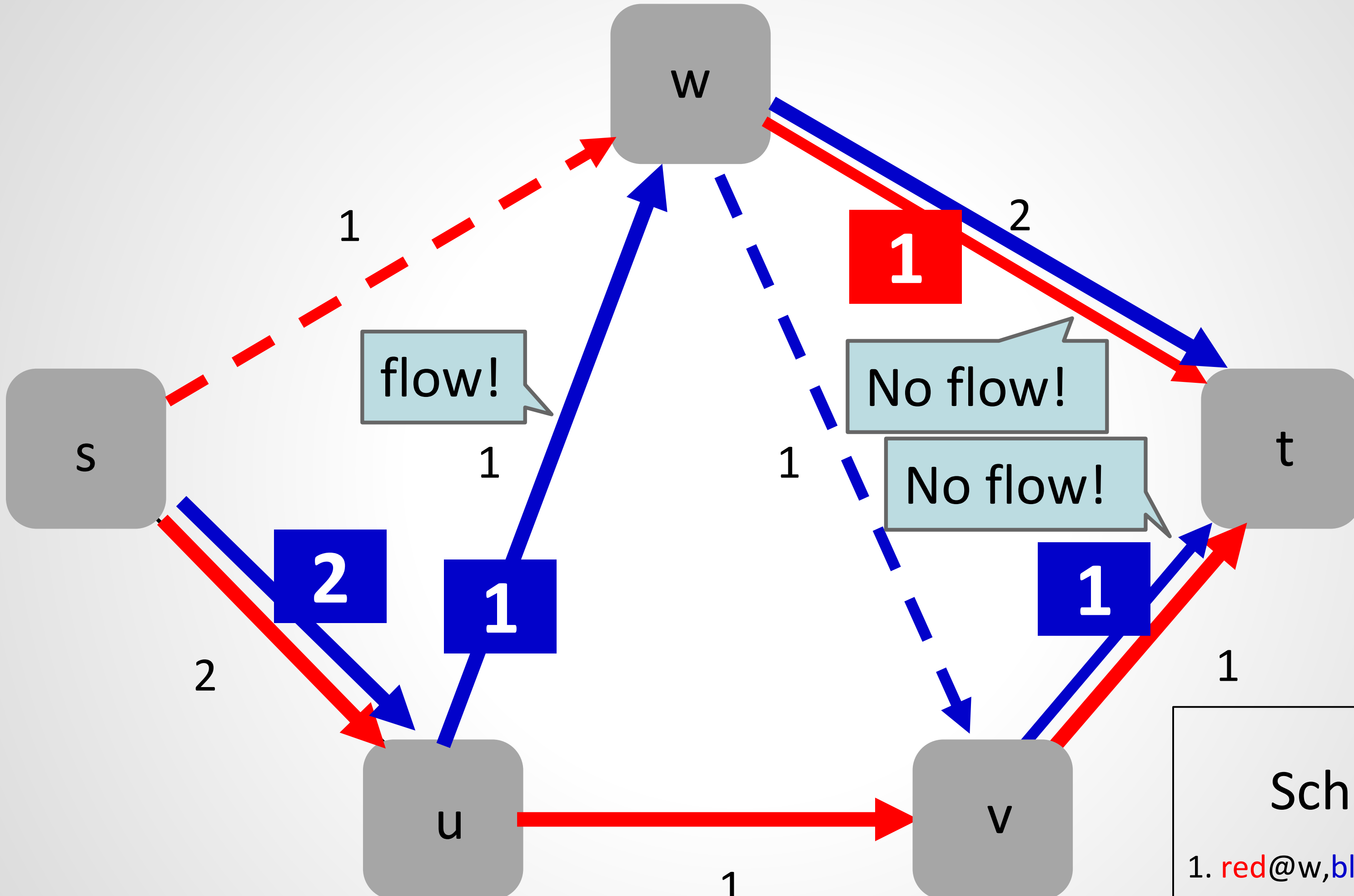


Round 1: prepare

Schedule:

1. red@w,blue@u,blue@v

What about capacity constraints?

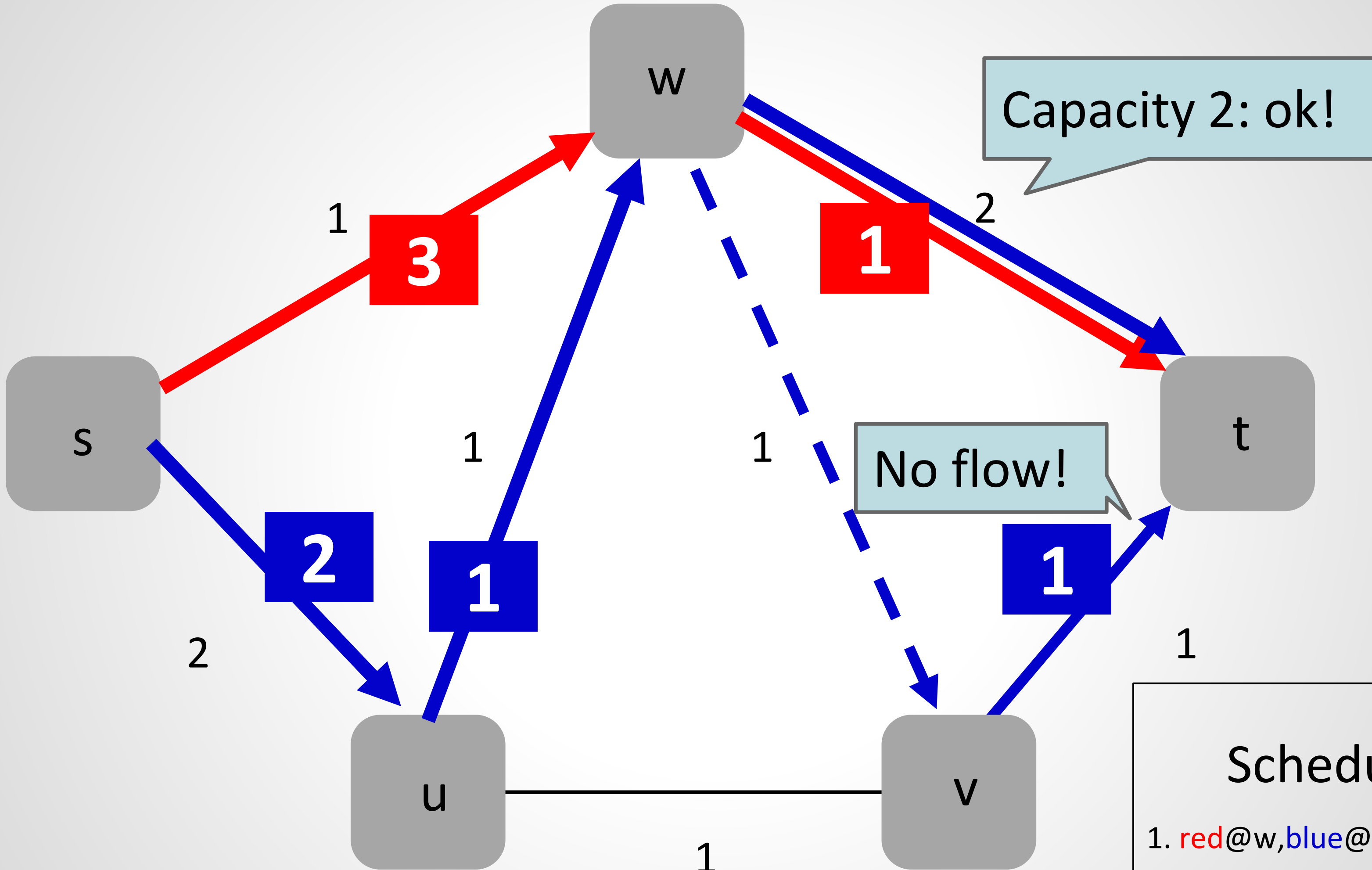


Schedule:

1. red@w,blue@u,blue@v
2. blue@s

Round 2

What about capacity constraints?

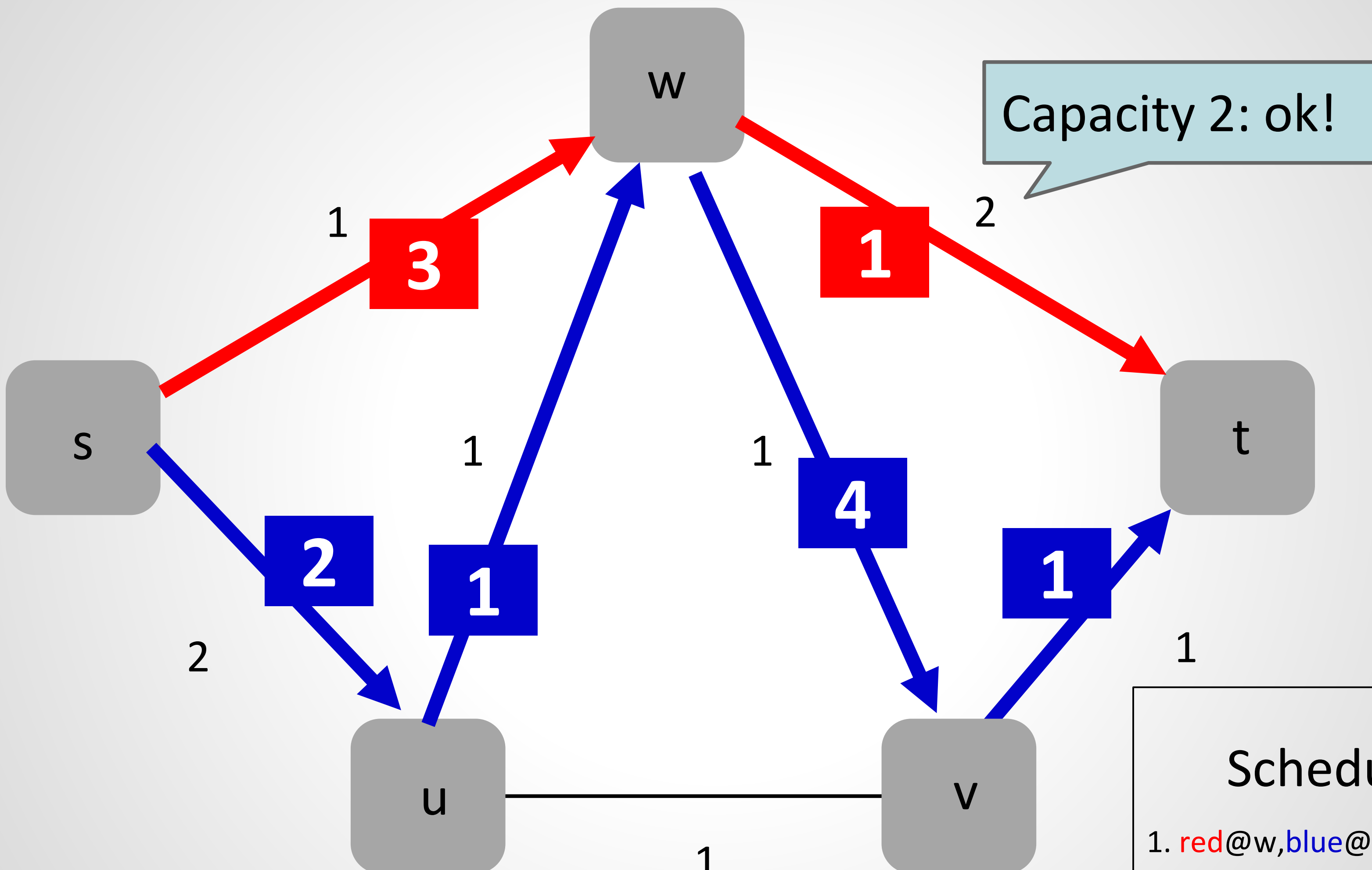


Schedule:

1. red@w,blue@u,blue@v
2. blue@s
3. red@s

Round 3

What about capacity constraints?

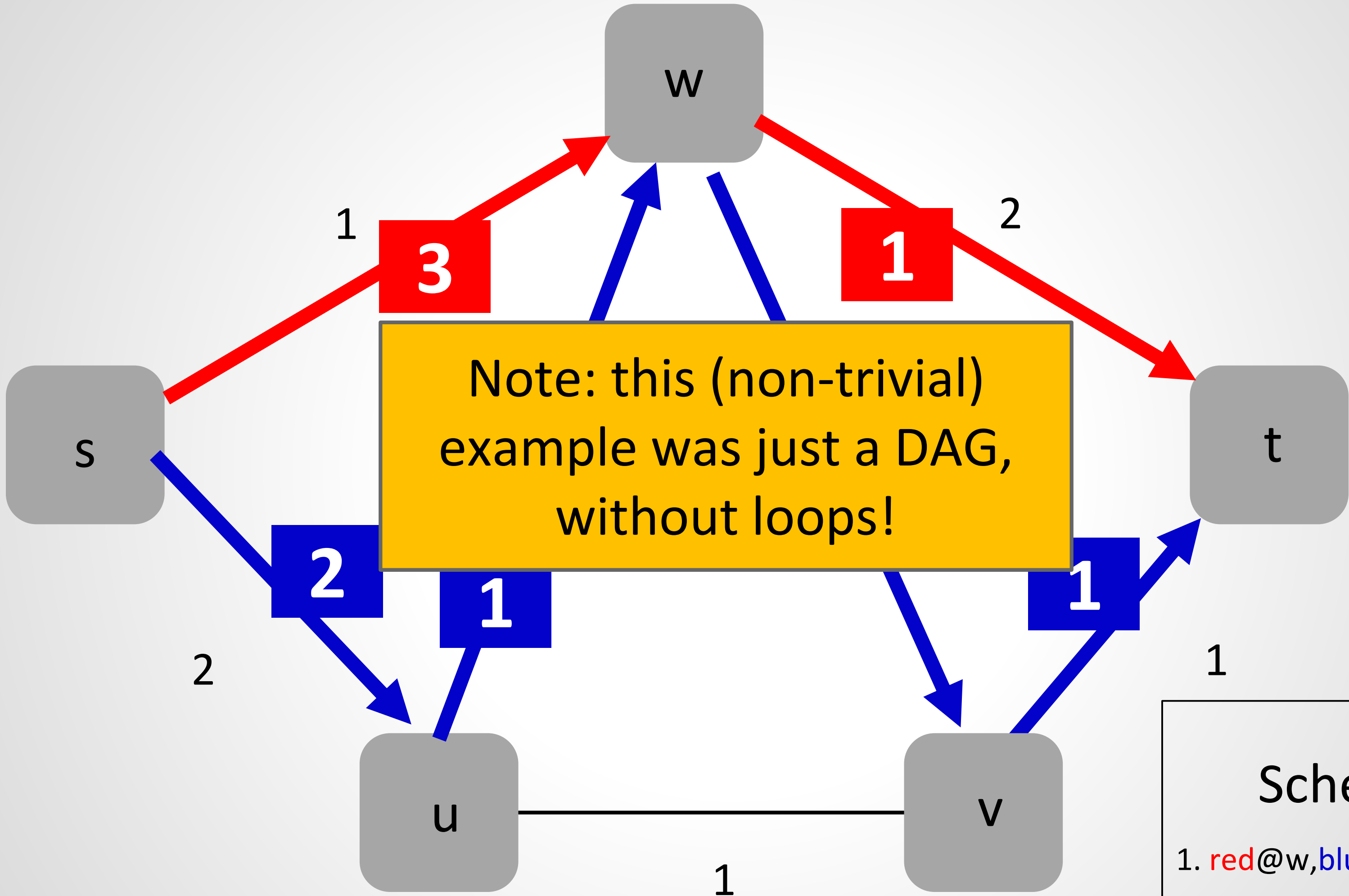


Schedule:

1. red@w,blue@u,blue@v
2. blue@s
3. red@s
4. blue@w

Round 4

What about capacity constraints?



Note: this (non-trivial) example was just a DAG, without loops!

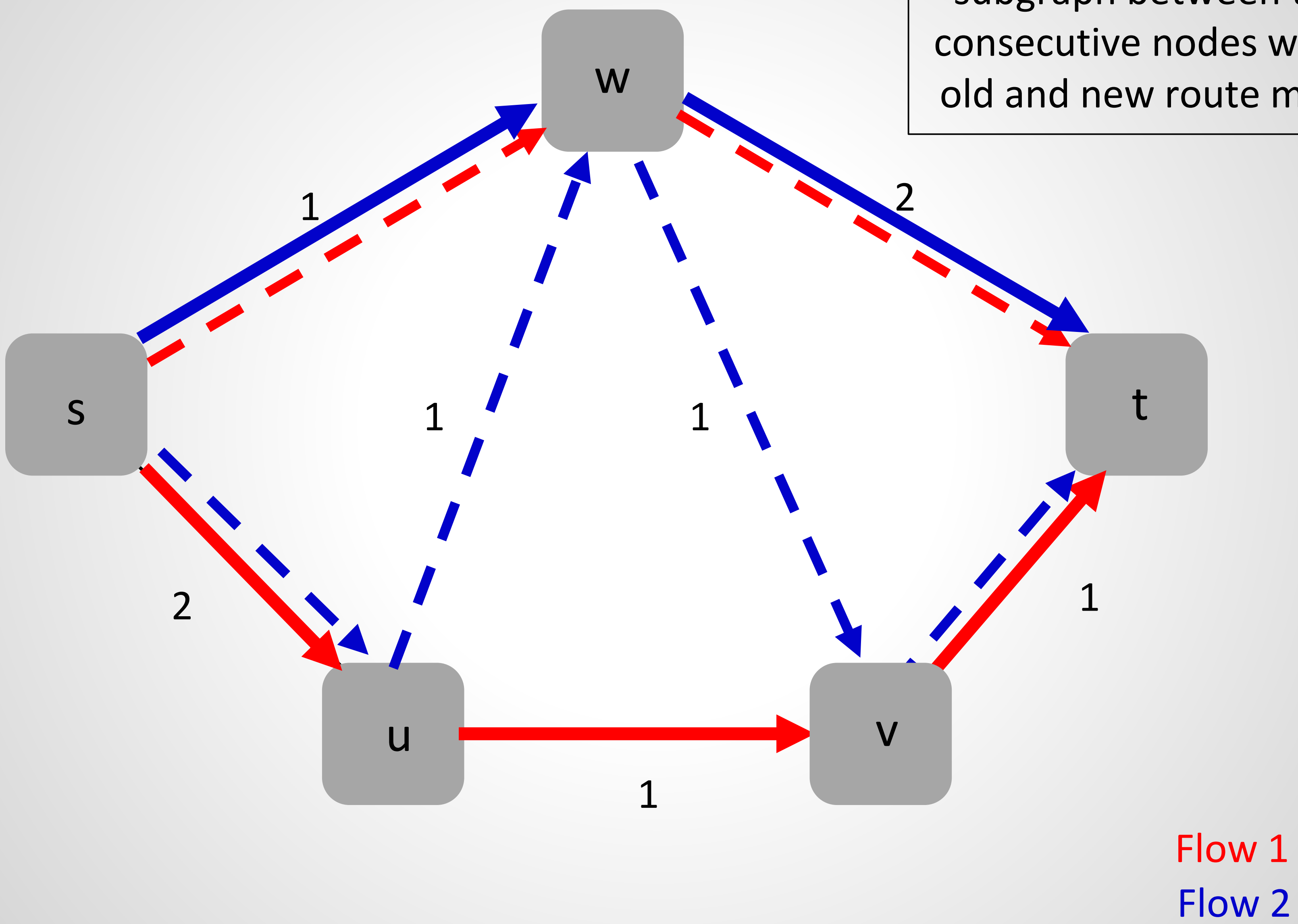
Schedule:

1. red@w,blue@u,blue@v
2. blue@s
3. red@s
4. blue@w

Round 4

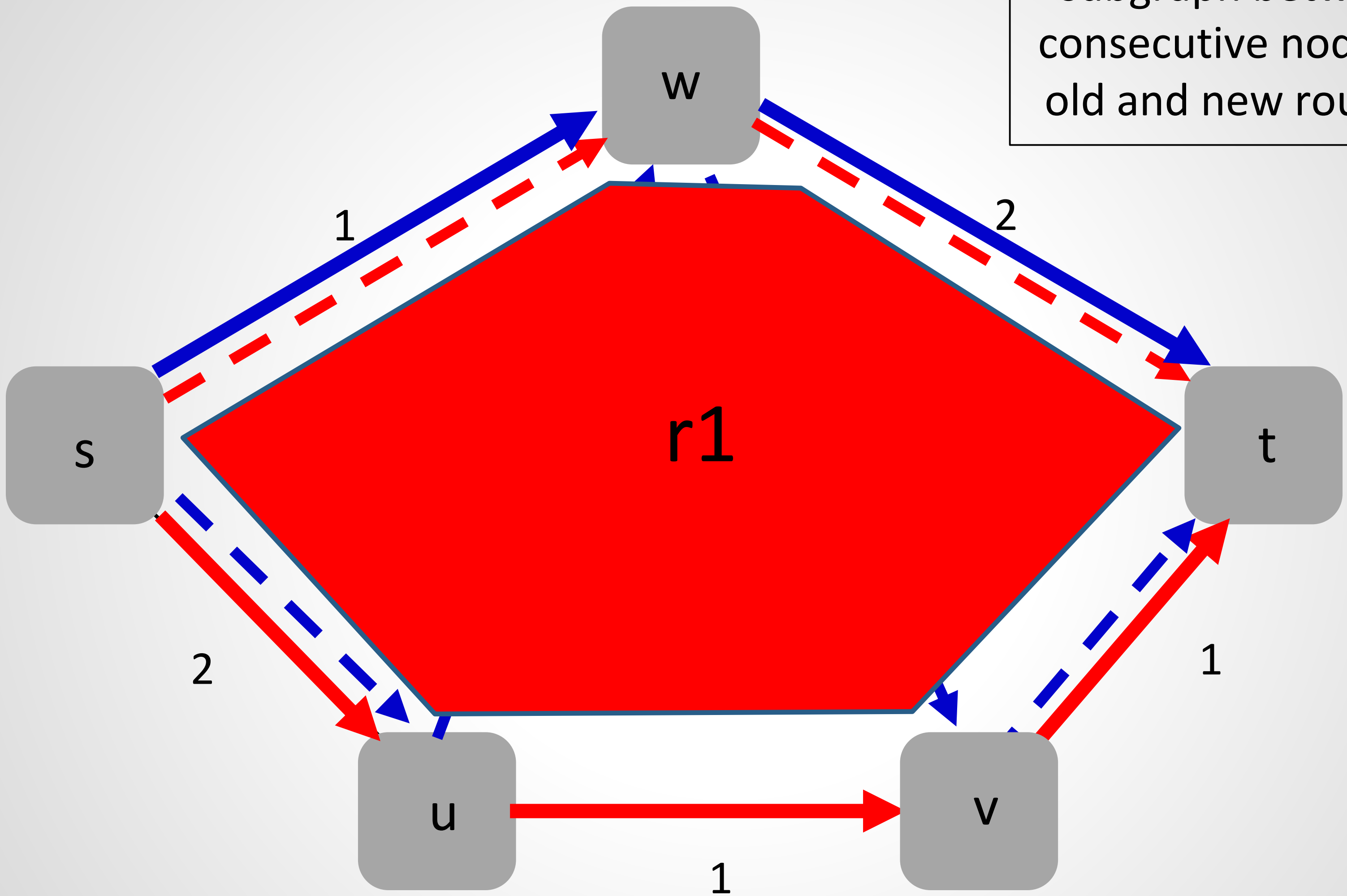
Block Decomposition of DAGs

Block for a given flow: subgraph between two consecutive nodes where old and new route meet.



Block Decomposition of DAGs

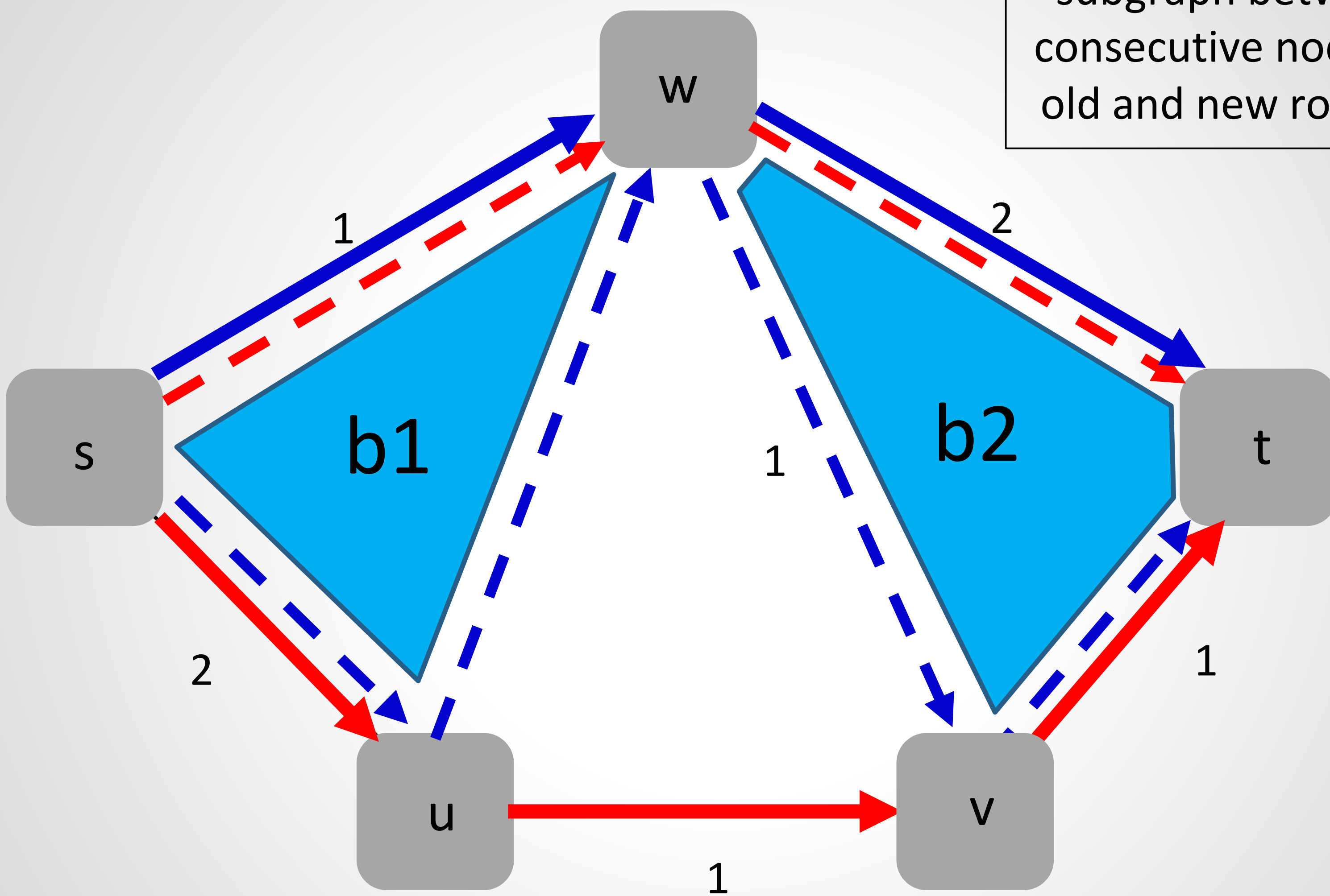
Block for a given flow: subgraph between two consecutive nodes where old and new route meet.



Just one red block: r1

Block Decomposition of DAGs

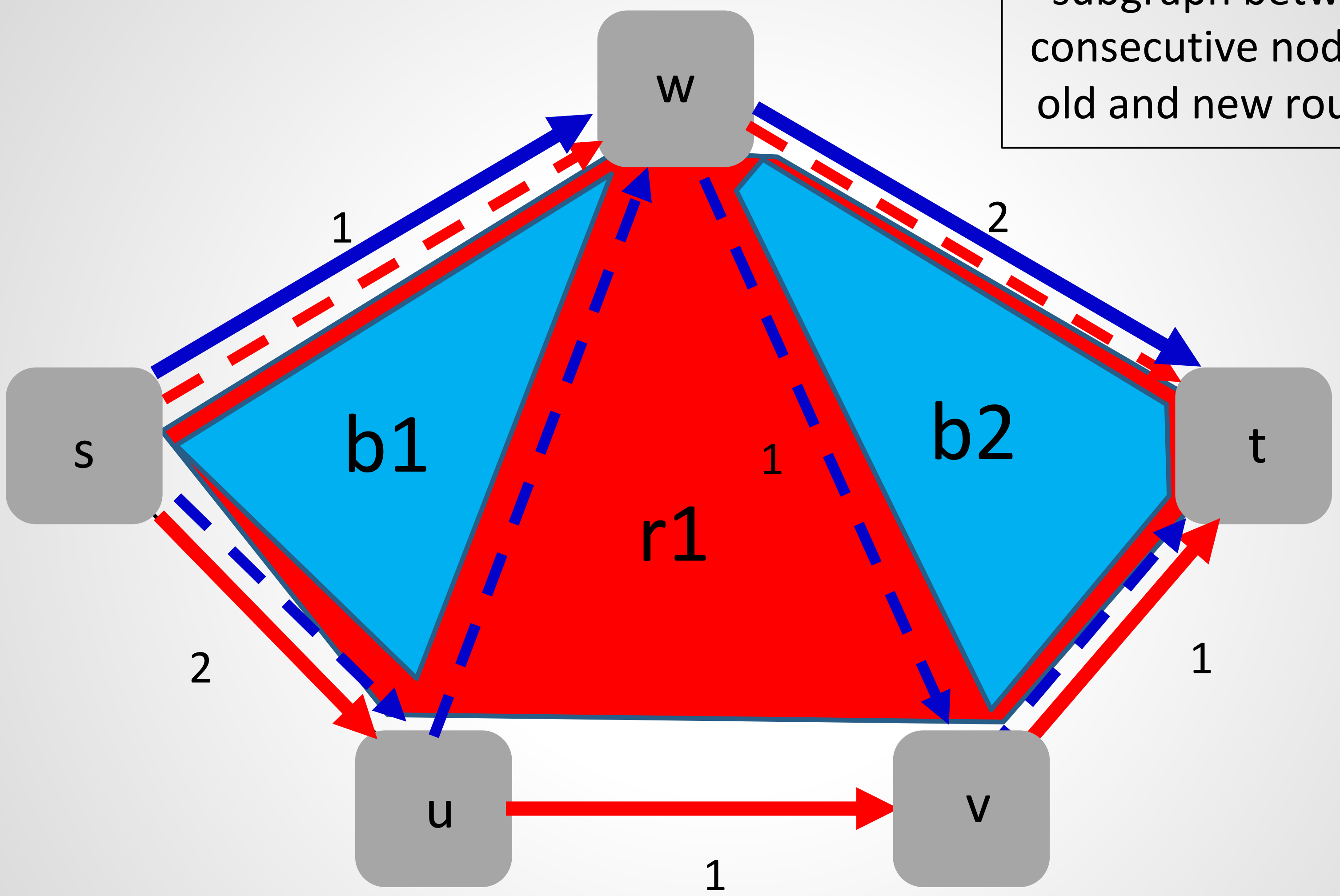
Block for a given flow: subgraph between two consecutive nodes where old and new route meet.



Two blue blocks: b1 and b2

Block Decomposition of DAGs

Block for a given flow: subgraph between two consecutive nodes where old and new route meet.



Dependencies: update b2 after r1 after b1.

Algorithms and Properties

- For k=2 flows
  - Using dependency graph of DAG block decomposition: feasible update exists if and only if cycle-free dependency
  - Also directly yields optimal number of rounds!
- For general k flows
  - Harder: We need a weaker notion of dependency graph
  - Only feasibility for constant k in polynomial-time
  - For general k, NP-hard
- Not much more is known so far
  - NP-hard on general networks already for 2 flows

Algorithms and Properties

- For k=2 flows
  - Using dependency graph of DAG block decomposition: feasible update exists if and only if cycle-free dependency
  - Also directly yields optimal number of rounds!
- Fo
  - ph
  - Only feasibility for constant k in polynomial time
  - For general k, NP-hard
- Not much more is known so far
  - NP-hard on general networks already for 2 flows

Essentially a combinatorial reconfiguration problem!

Many Open Algorithmic Problems

- Complexity of scheduling (weak) loop-free updates? What about approximations?
- Congestion-free update algorithms beyond DAGs?
- What about multiple waypoints?
- Related to Reconfiguration Graph Theory!

Further Reading

[Can't Touch This: Consistent Network Updates for Multiple Policies]
Szymon Dudycz, Arne Ludwig, and Stefan Schmid.
46th IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Toulouse, France, June 2016.

[Transiently Secure Network Updates]
Arne Ludwig, Szymon Dudycz, Matthias Rost, and Stefan Schmid.
42nd ACM **SIGMETRICS**, Antibes Juan-les-Pins, France, June 2016.

[Scheduling Loop-free Network Updates: It's Good to Relax!]
Arne Ludwig, Jan Marcinkowski, and Stefan Schmid.
ACM Symposium on Principles of Distributed Computing (**PODC**), Donostia-San Sebastian, Spain, July 2015.

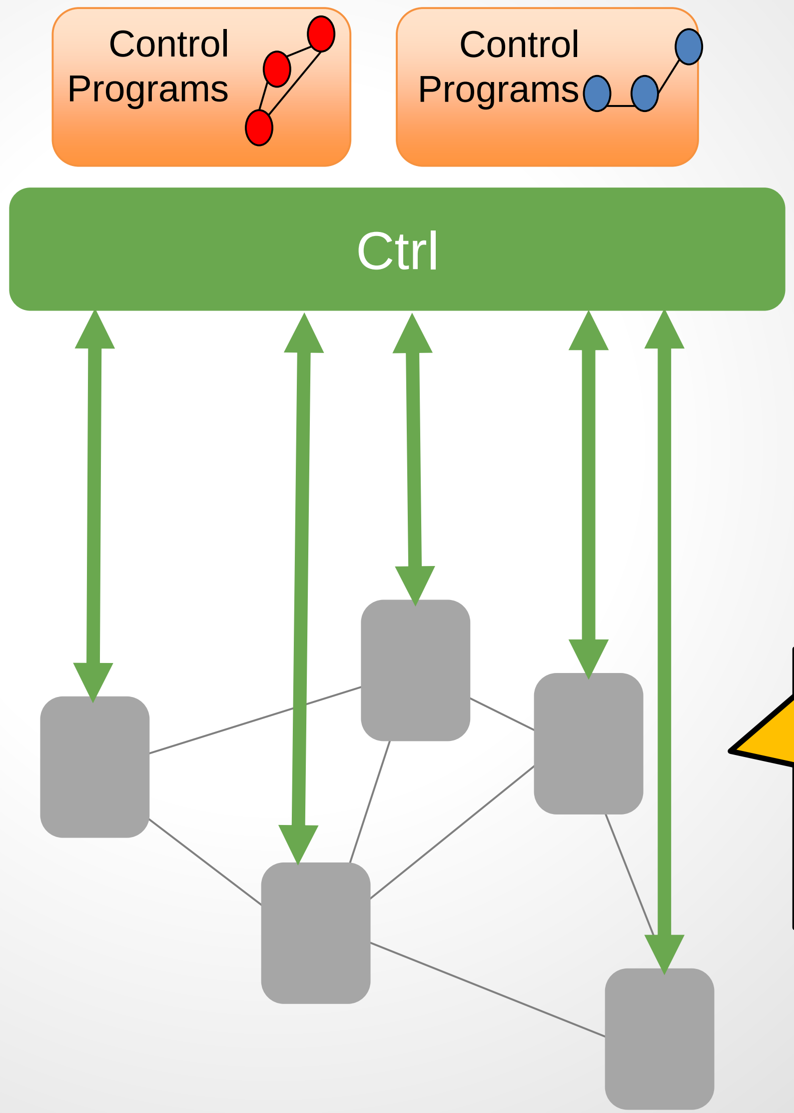
[Congestion-Free Rerouting of Flows on DAGs]
Saeed Akhoondian Amiri, Szymon Dudycz, Stefan Schmid, and Sebastian Wiederrecht. ArXiv Technical Report, November 2016.

Algorithmic Problems in SDNs

Applications and Control Plane

... and regarding decoupling / inter-connect!

Data Plane



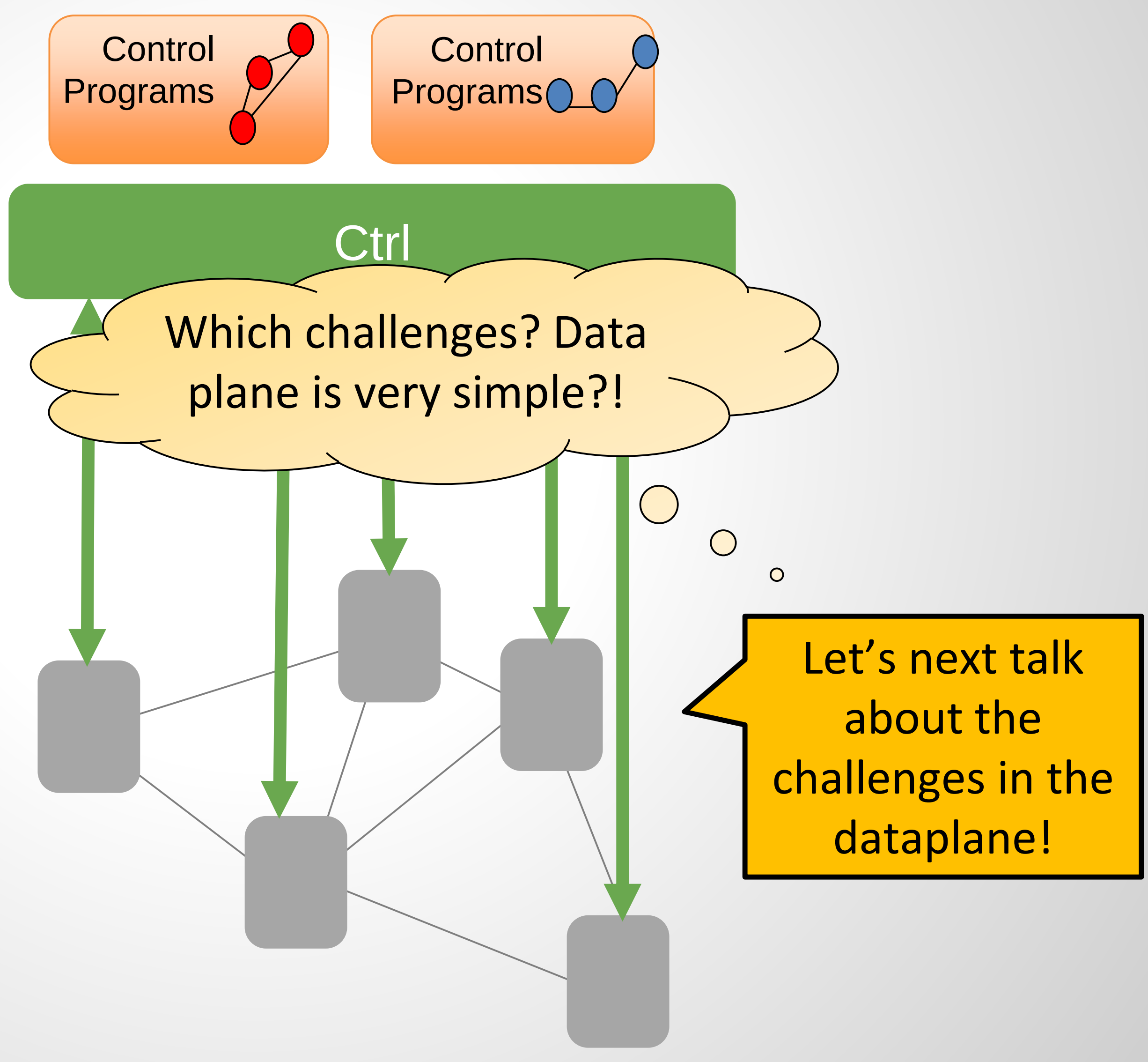
Let's next talk about the challenges in the dataplane!

Algorithmic Problems in SDNs

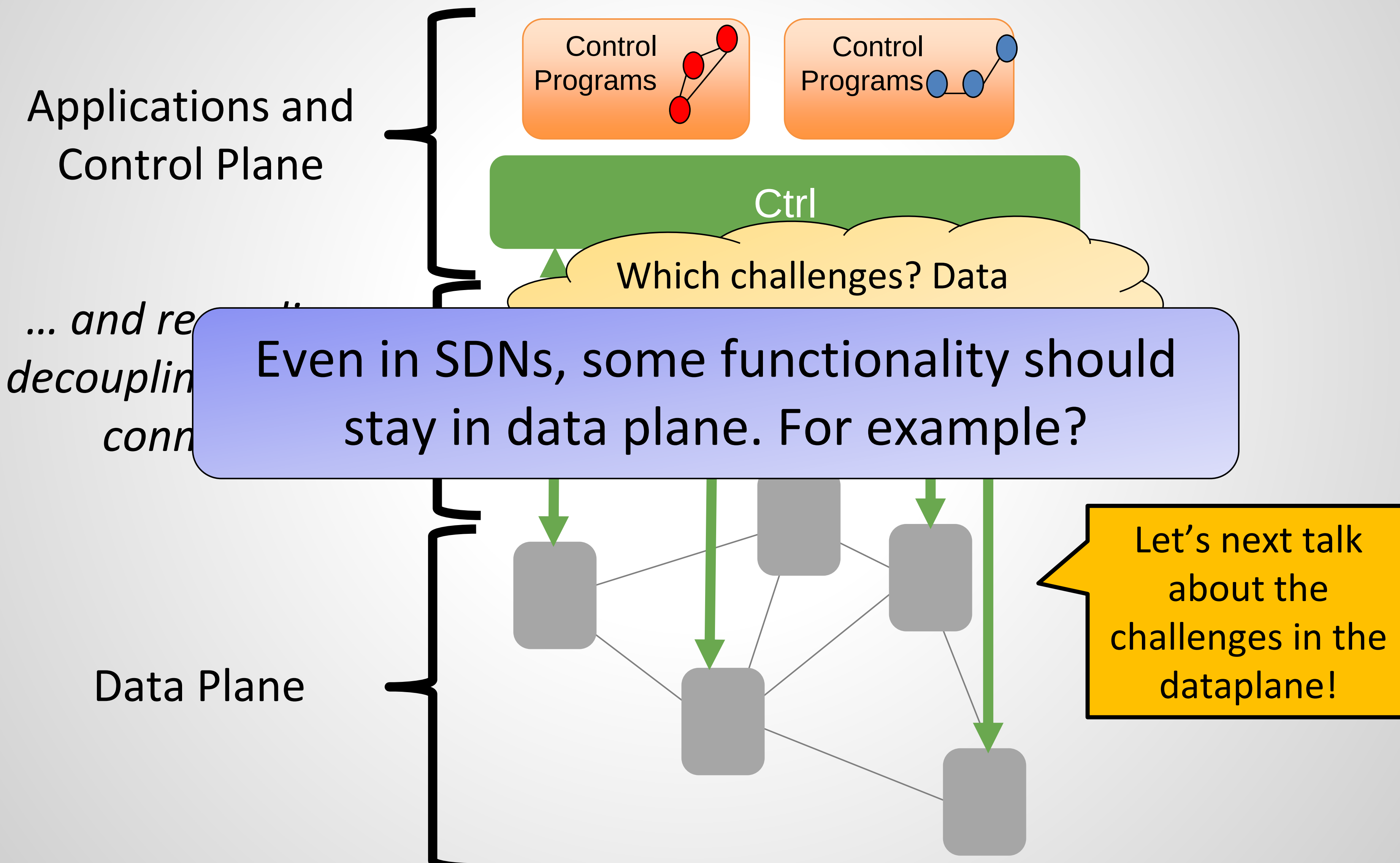
Applications and Control Plane

… and regarding decoupling / inter-connect!

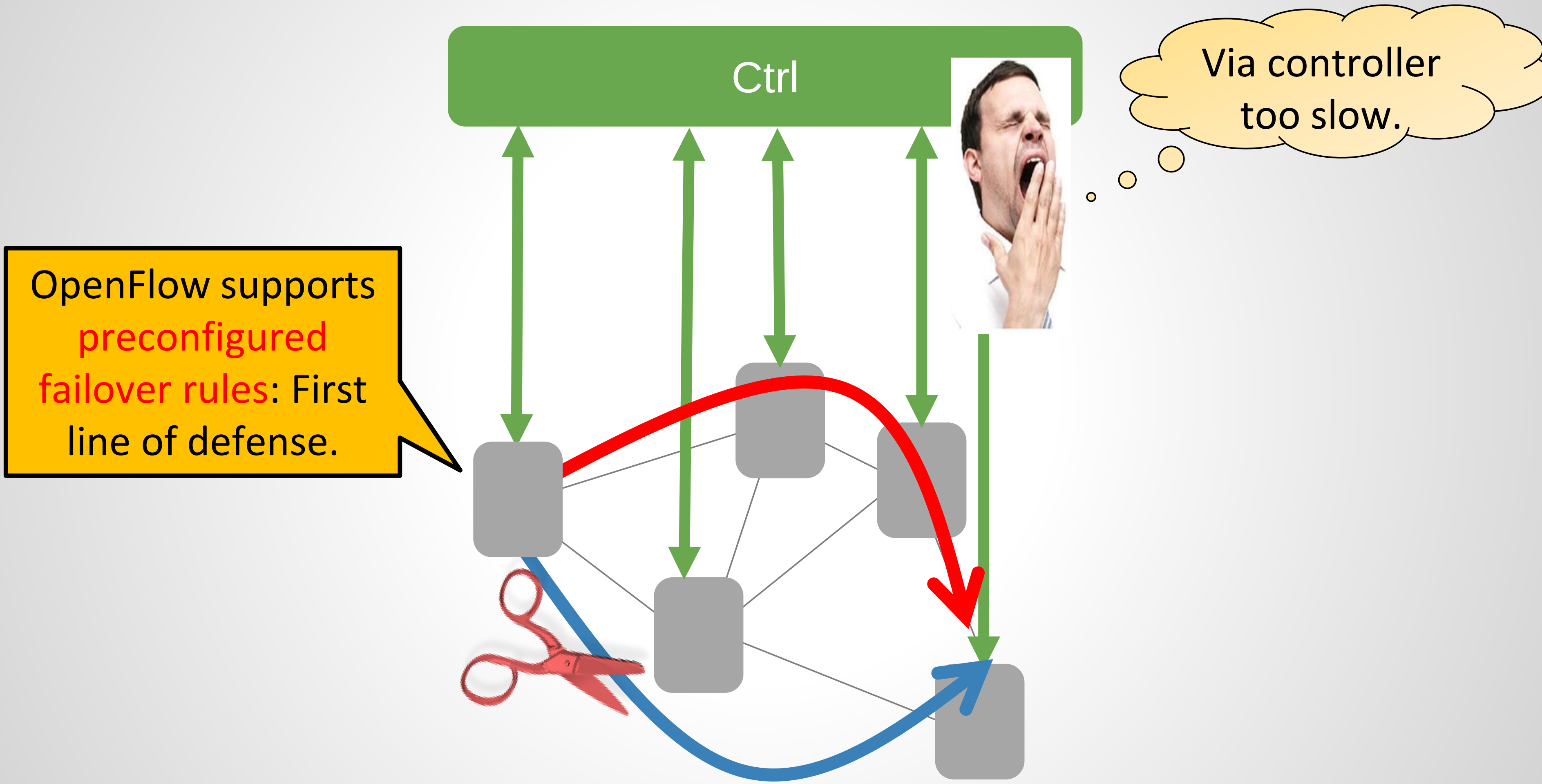
Data Plane



Algorithmic Problems in SDNs

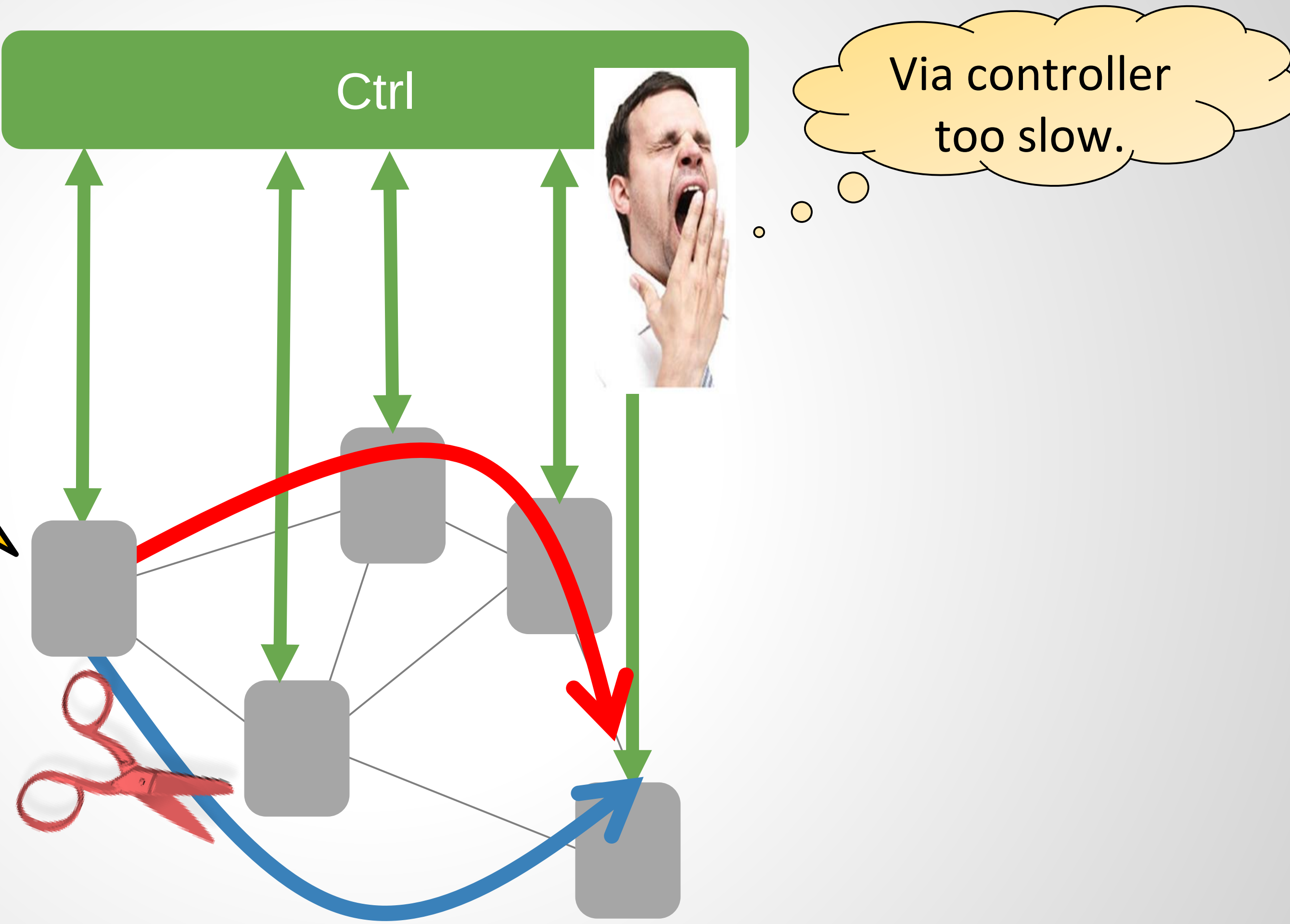


Should Stay in Data Plane: Local Fast Failover



Should Stay in Data Plane: Local Fast Failover

OpenFlow supports preconfigured failover rules: First line of defense.

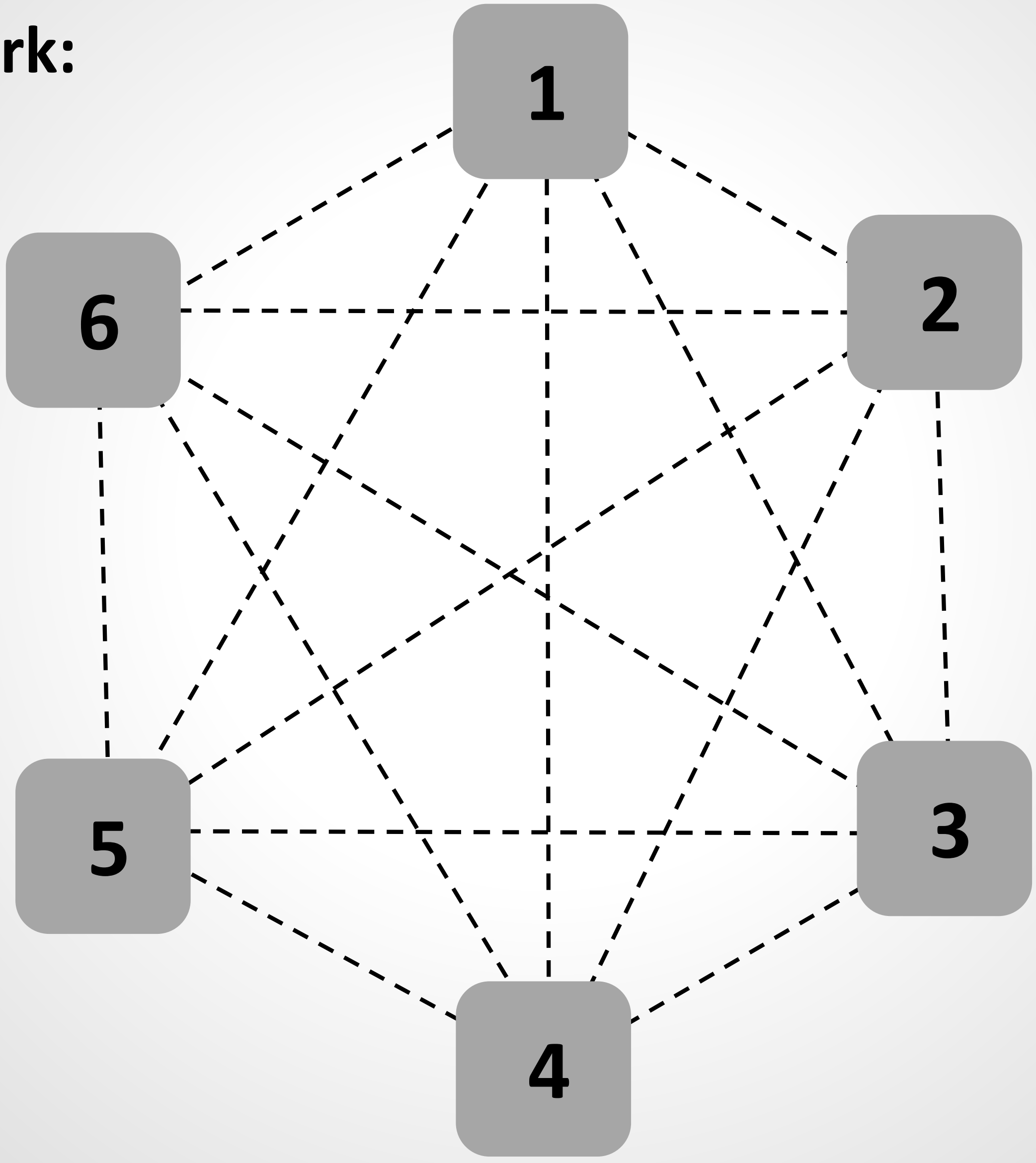


The Crux: **How to define conditional rules which have local failure knowledge only?**

Efficient Local Fast Failover: Non-Trivial Already in the Clique!

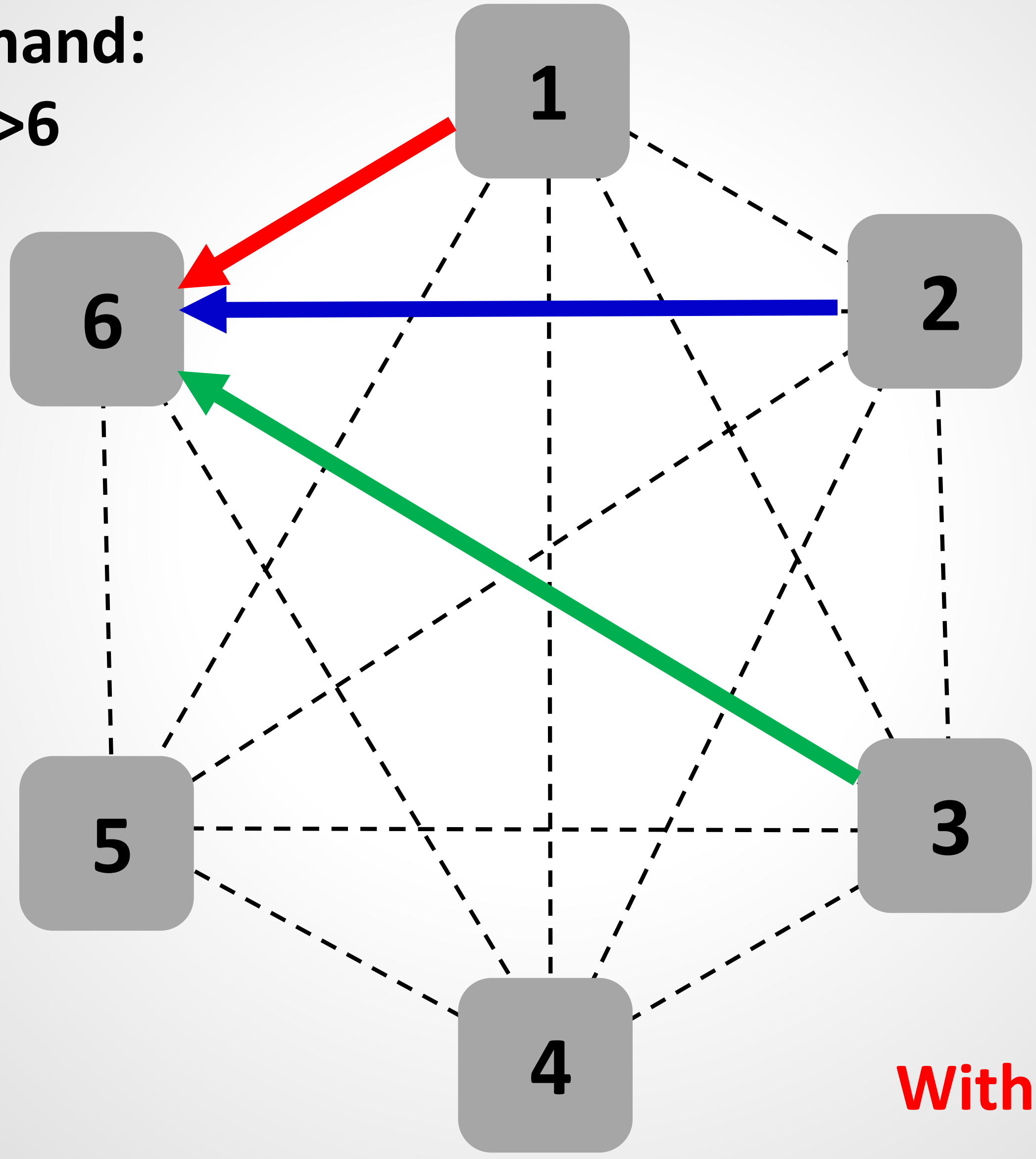
Local Fast Failover

The network:



Local Fast Failover

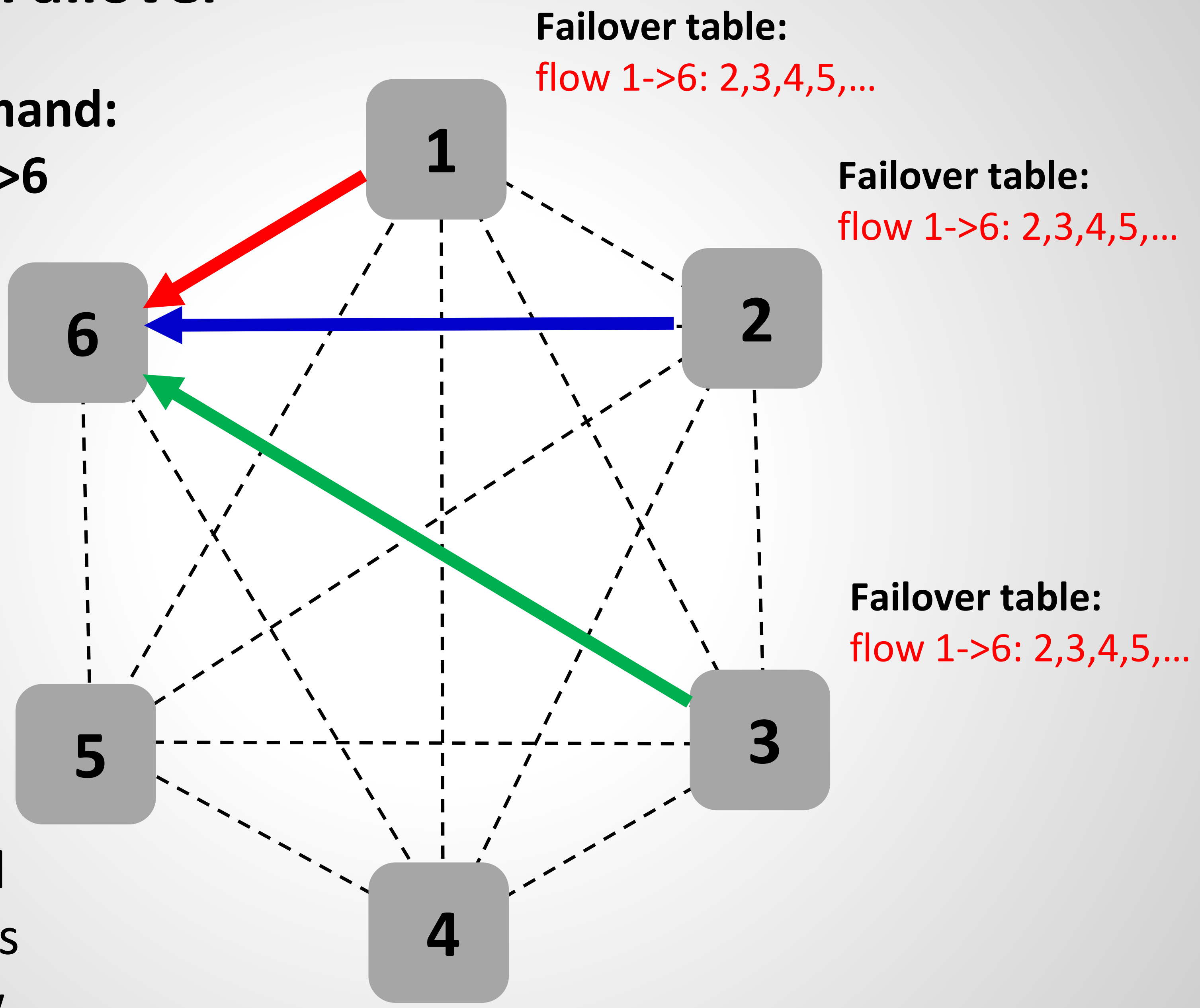
Traffic demand:
{1,2,3}->6



Without failures!

Local Fast Failover

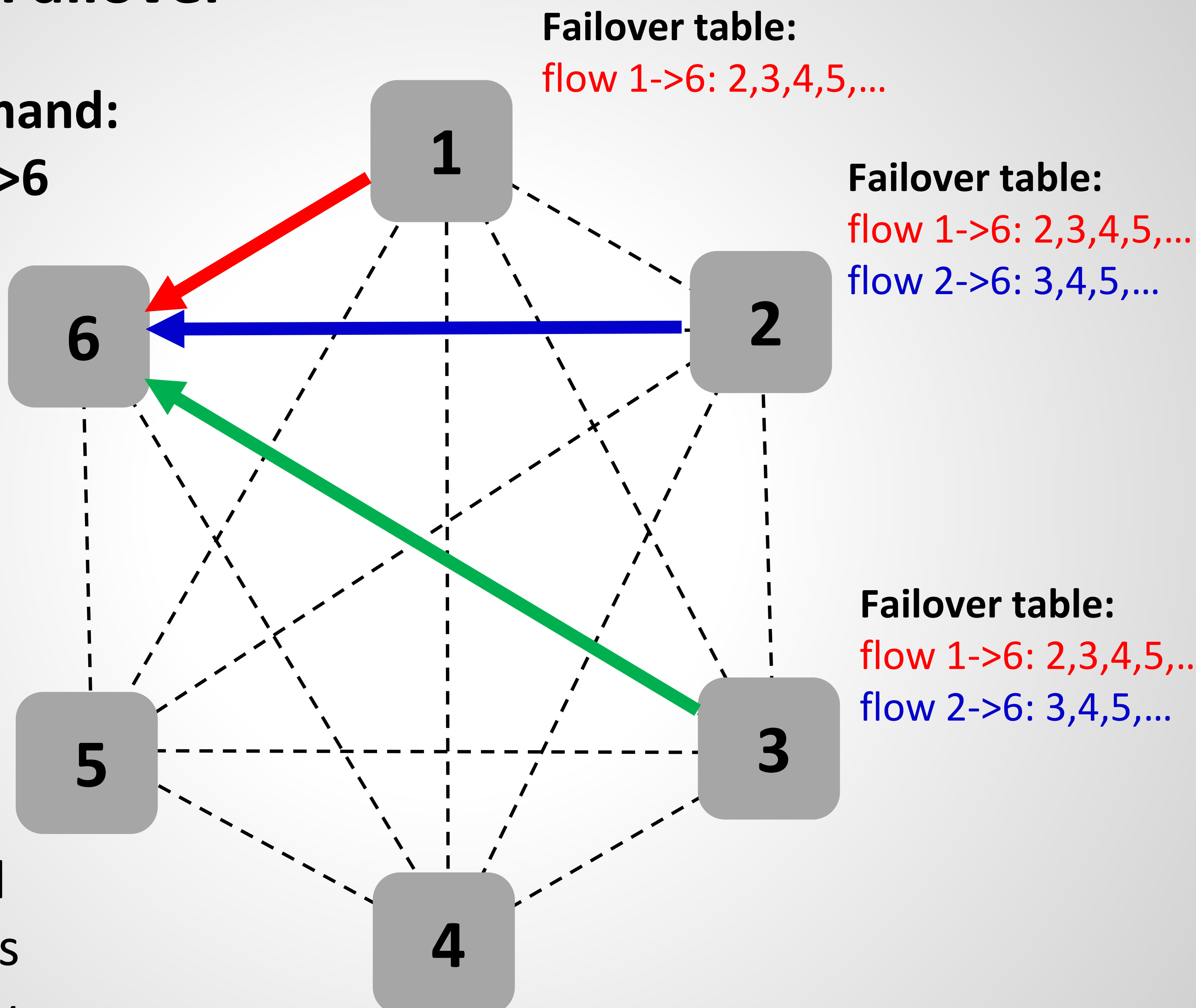
Traffic demand:
{1,2,3}->6



Preinstalled failover rules for red flow

Local Fast Failover

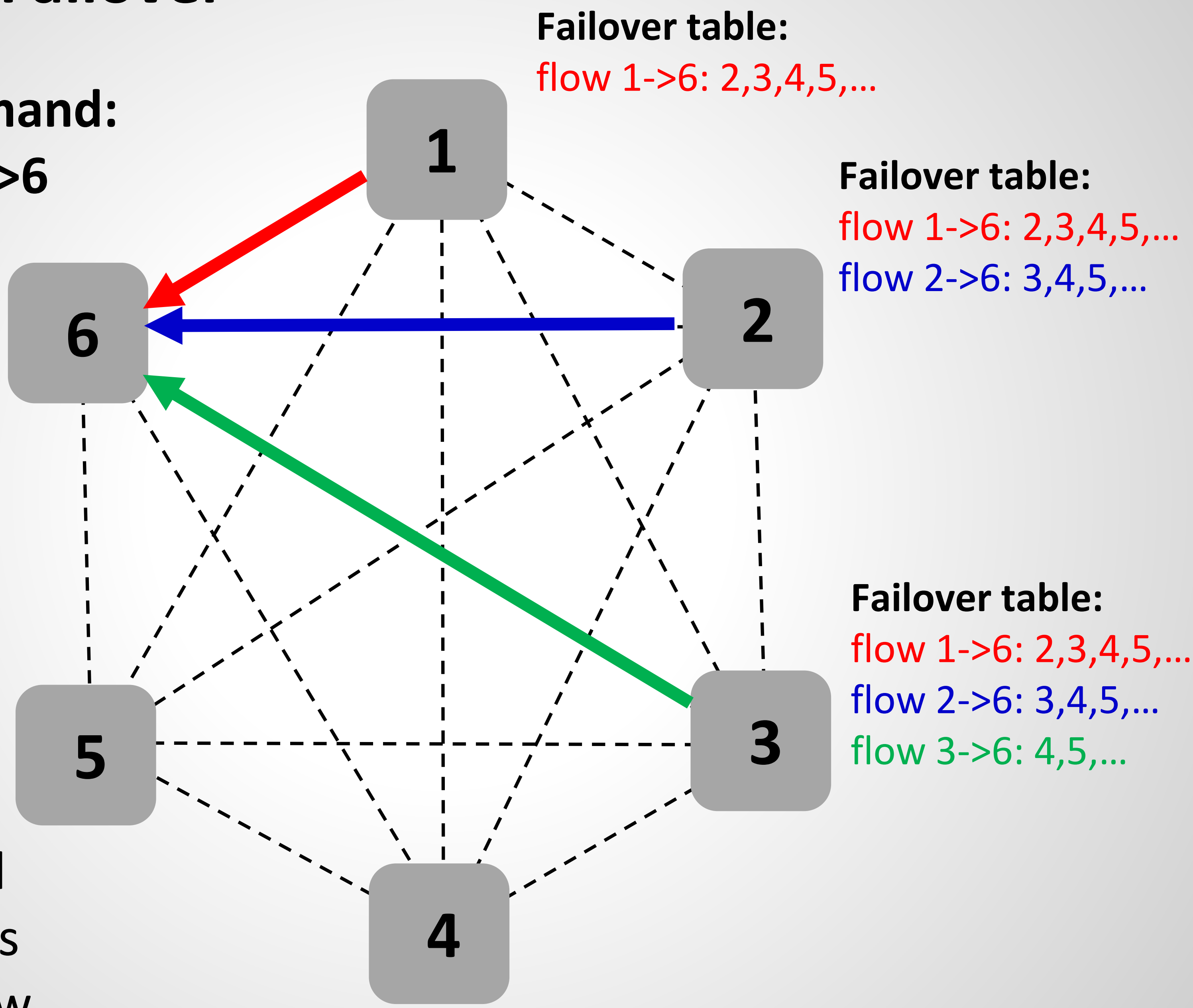
Traffic demand:
{1,2,3}->6



Preinstalled failover rules for blue flow

Local Fast Failover

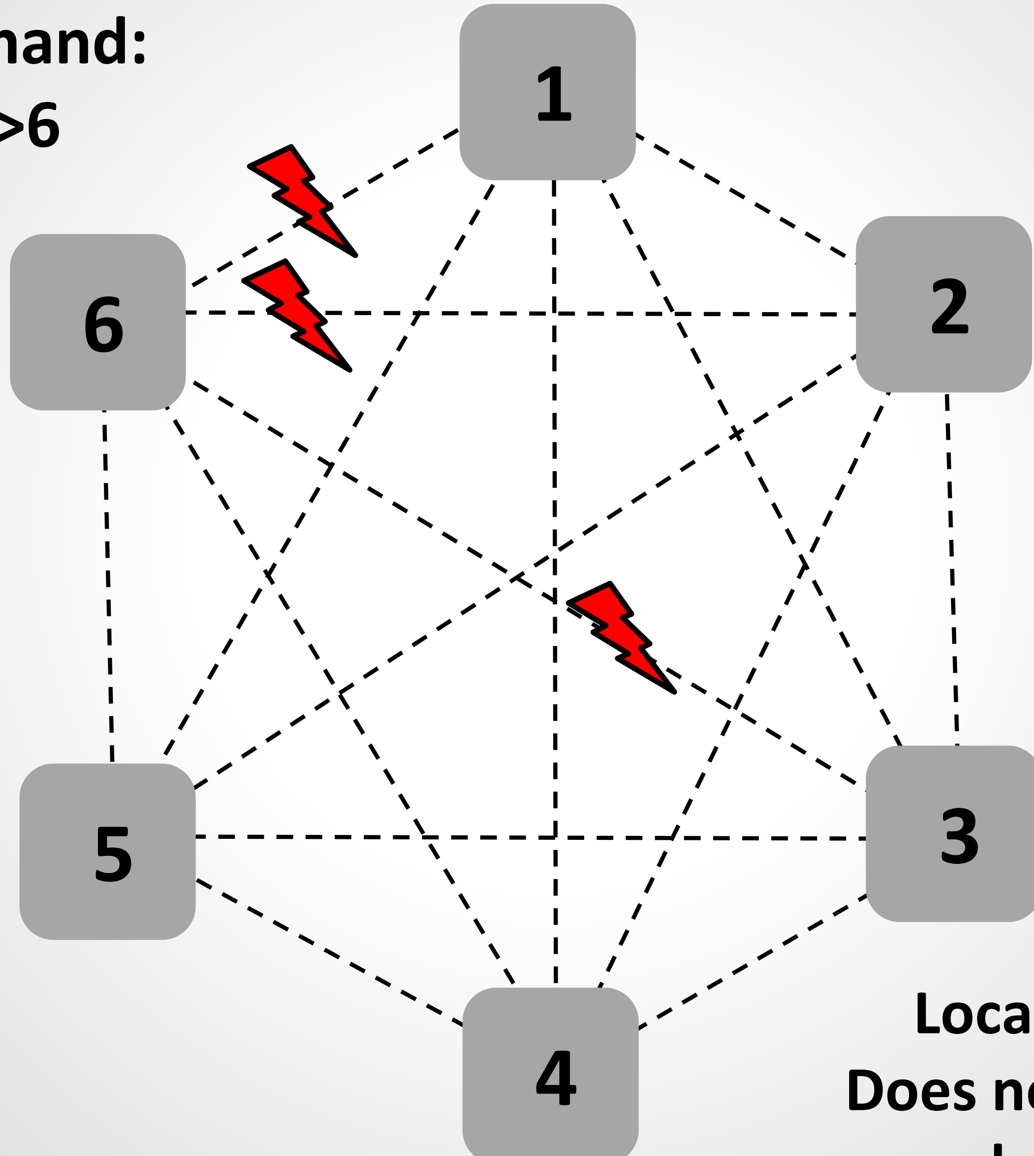
Traffic demand:
{1,2,3}->6



Preinstalled failover rules for green flow

Local Fast Failover

Traffic demand:
{1,2,3}->6

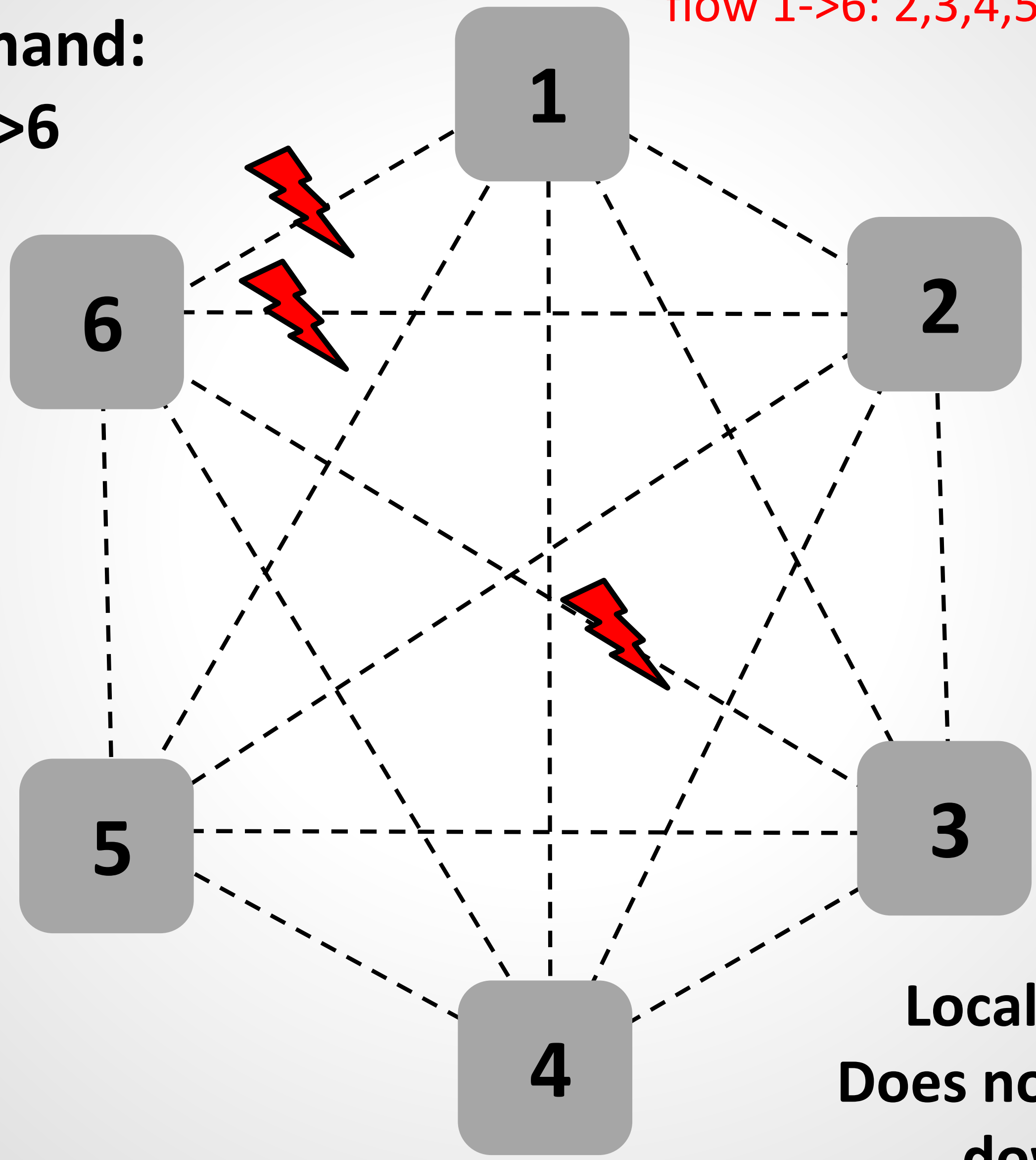


Local failover @1:
Does not know failures downstream!

Local Fast Failover

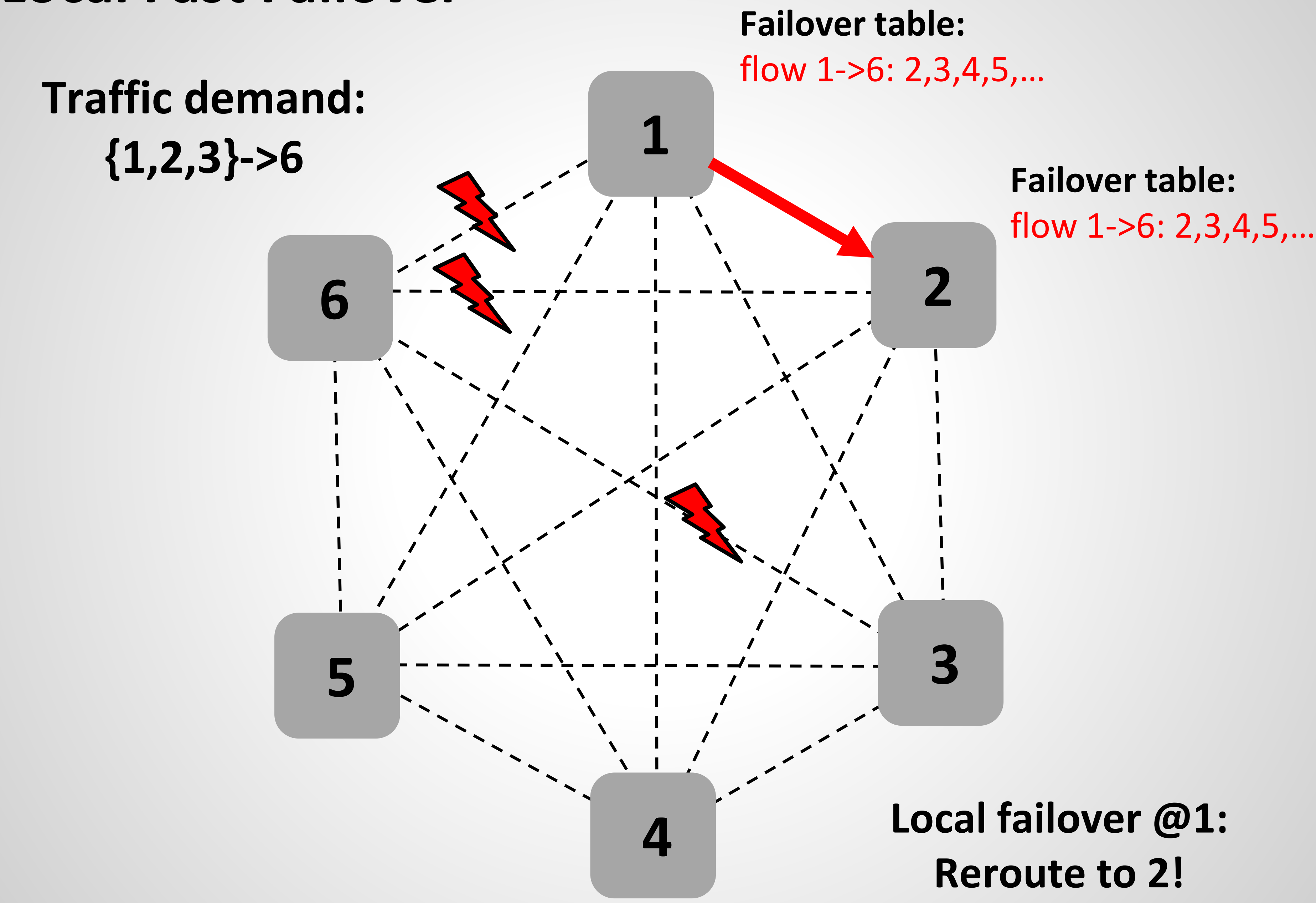
Traffic demand:
{1,2,3}->6

Failover table:
flow 1->6: 2,3,4,5,...



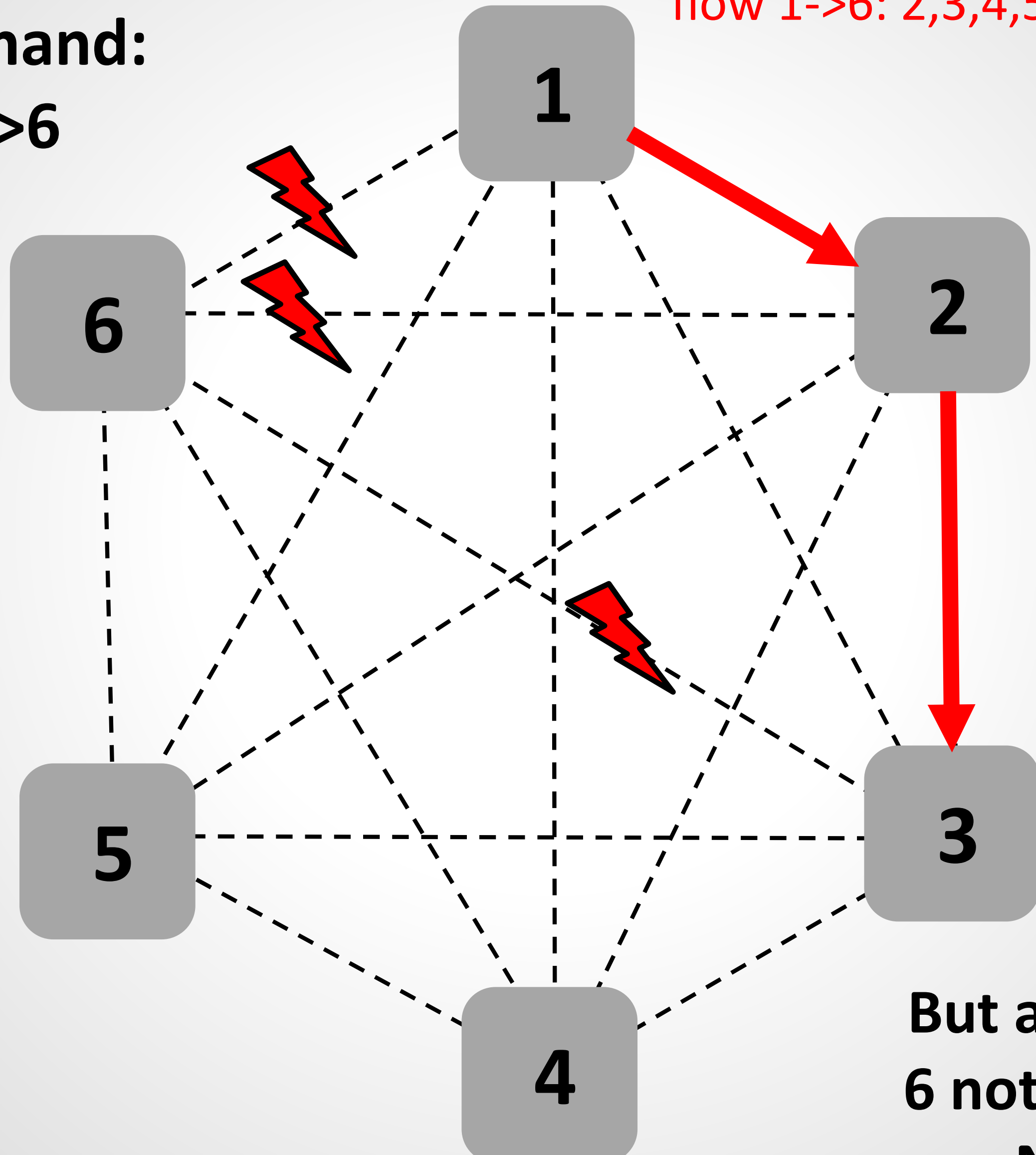
Local failover @1:
Does not know failures downstream!

Local Fast Failover



Local Fast Failover

Traffic demand:
{1,2,3}->6



Failover table:
flow 1->6: 2,3,4,5,...

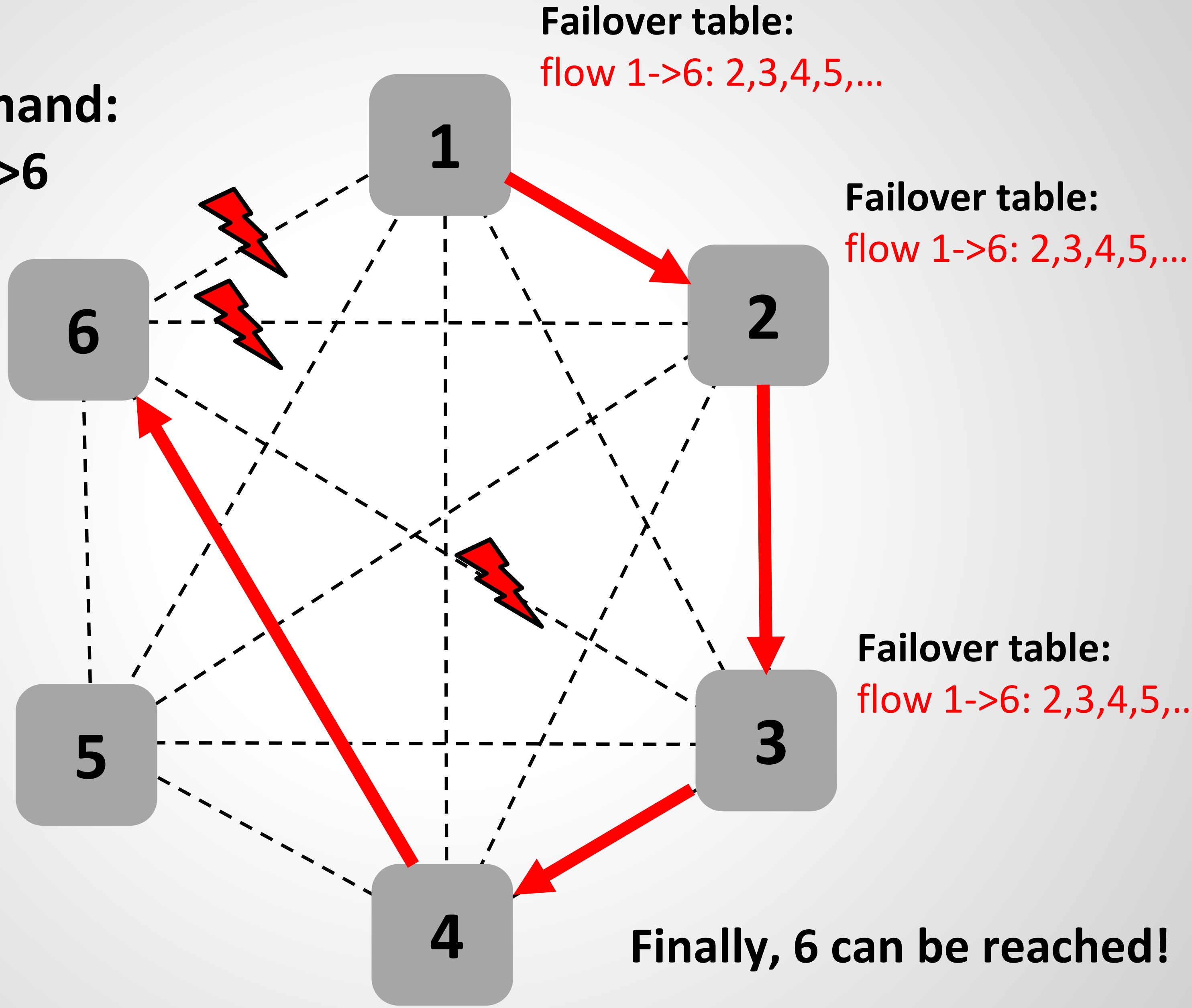
Failover table:
flow 1->6: 2,3,4,5,...

Failover table:
flow 1->6: 2,3,4,5,...

But also from 2:
6 not reachable.
Next: 3.

Local Fast Failover

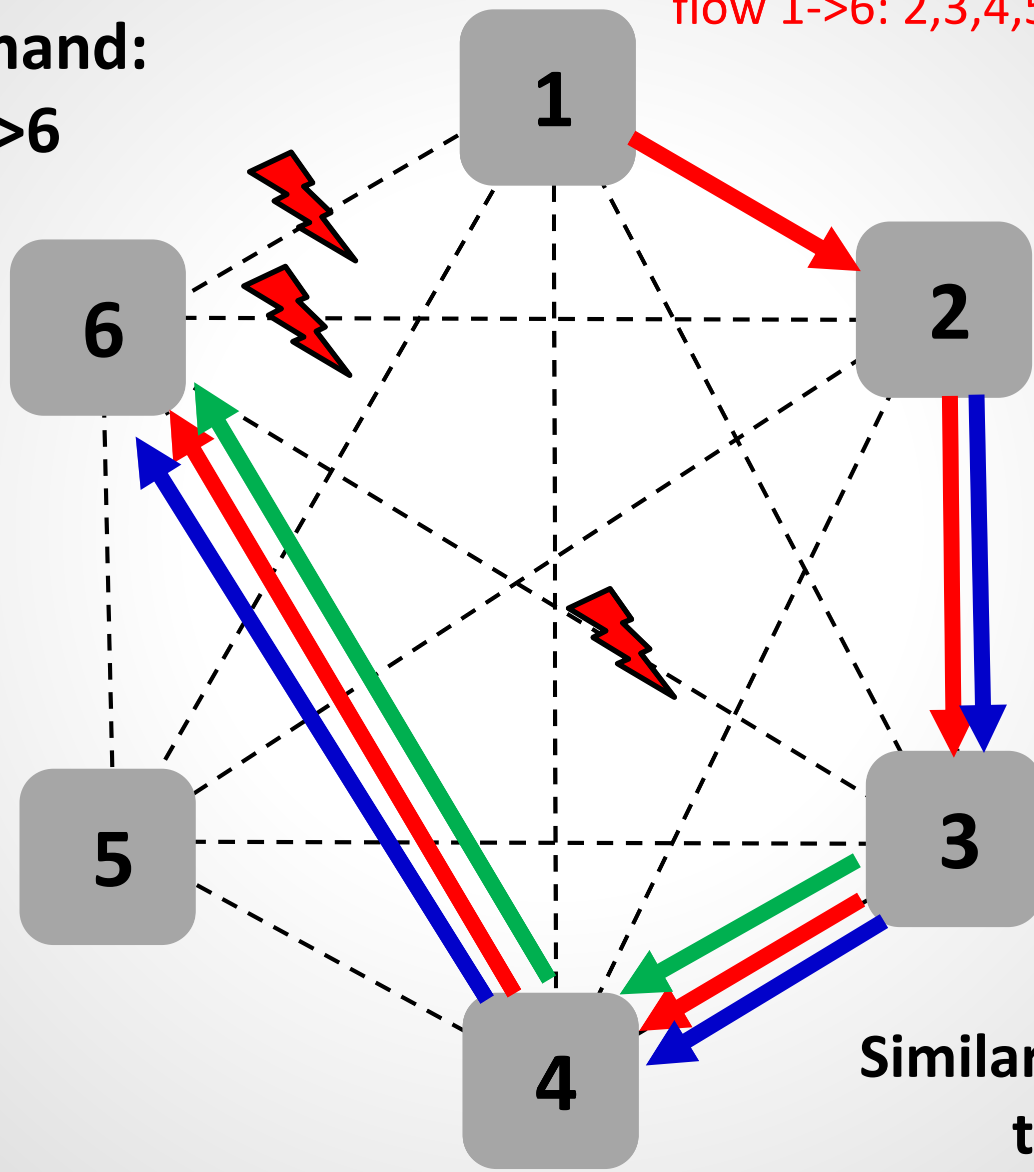
Traffic demand:
{1,2,3}->6



Finally, 6 can be reached!

Local Fast Failover

Traffic demand:
{1,2,3}->6



Failover table:
flow 1->6: 2,3,4,5,…

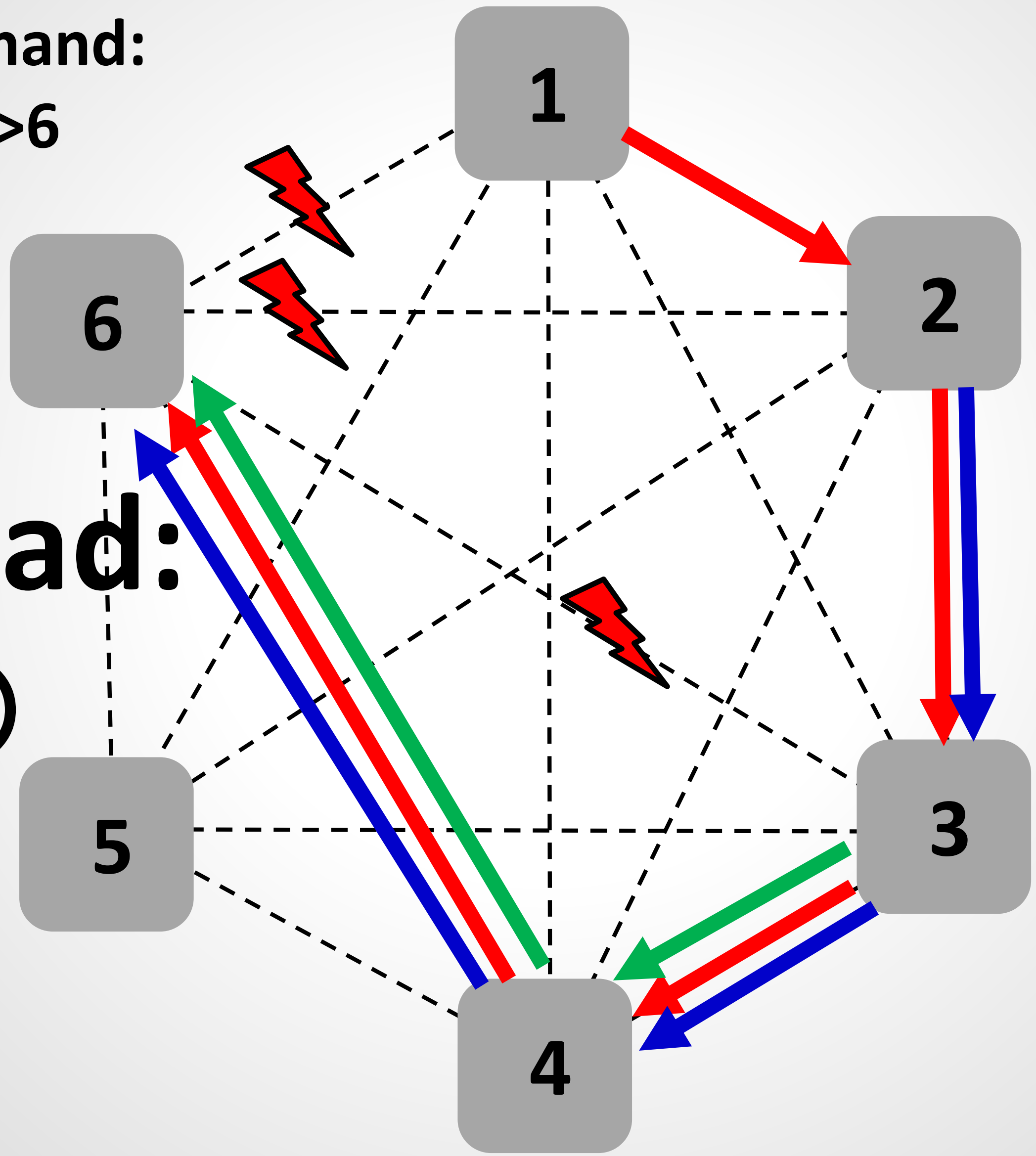
Failover table:
flow 1->6: 2,3,4,5,…
flow 2->6: 3,4,5,…

Failover table:
flow 1->6: 2,3,4,5,…
flow 2->6: 3,4,5,…
flow 3->6: 4,5,…

Similarly for the other two flows.

Local Fast Failover

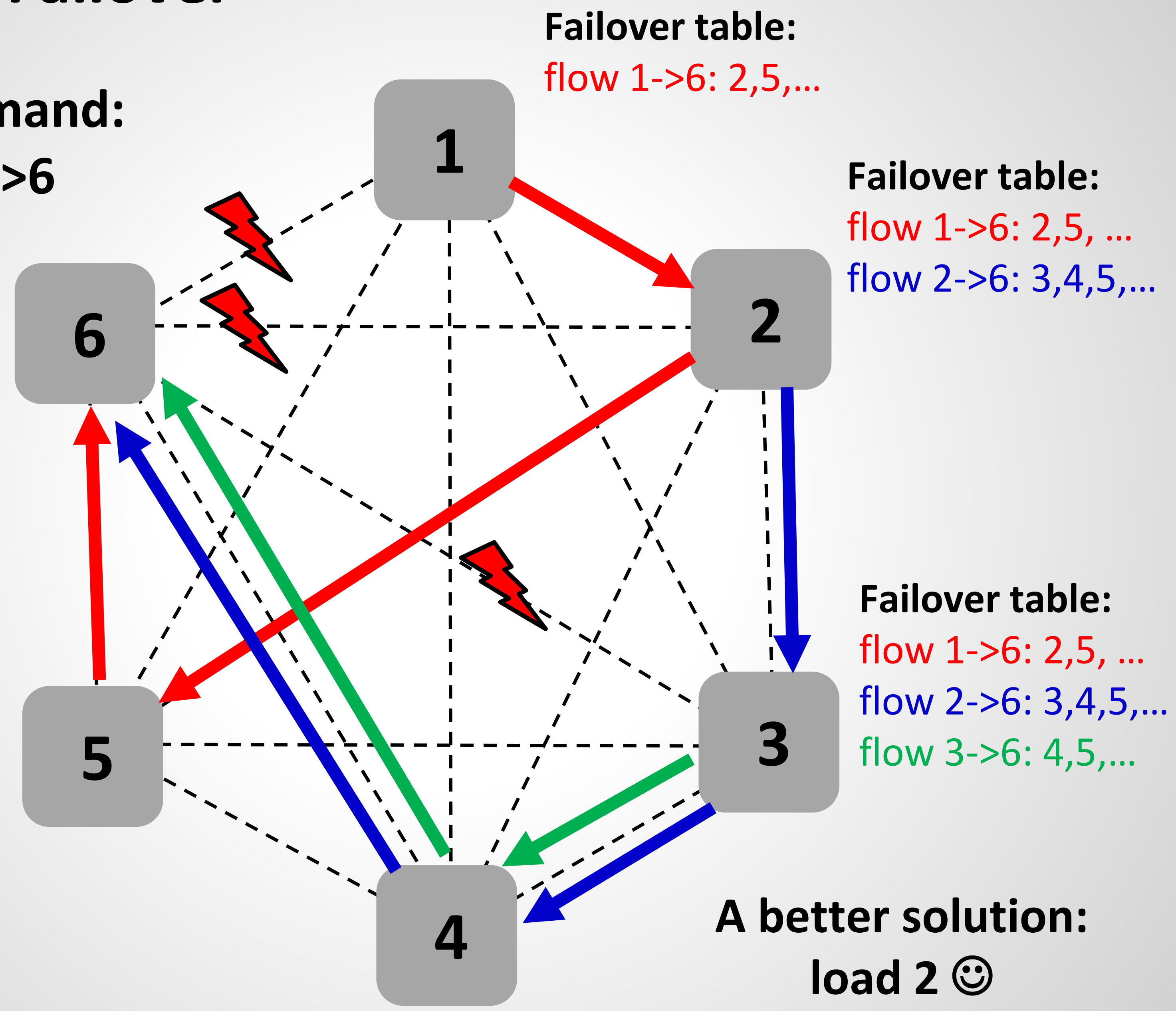
Traffic demand:
{1,2,3}->6



Max load:
3 ☹

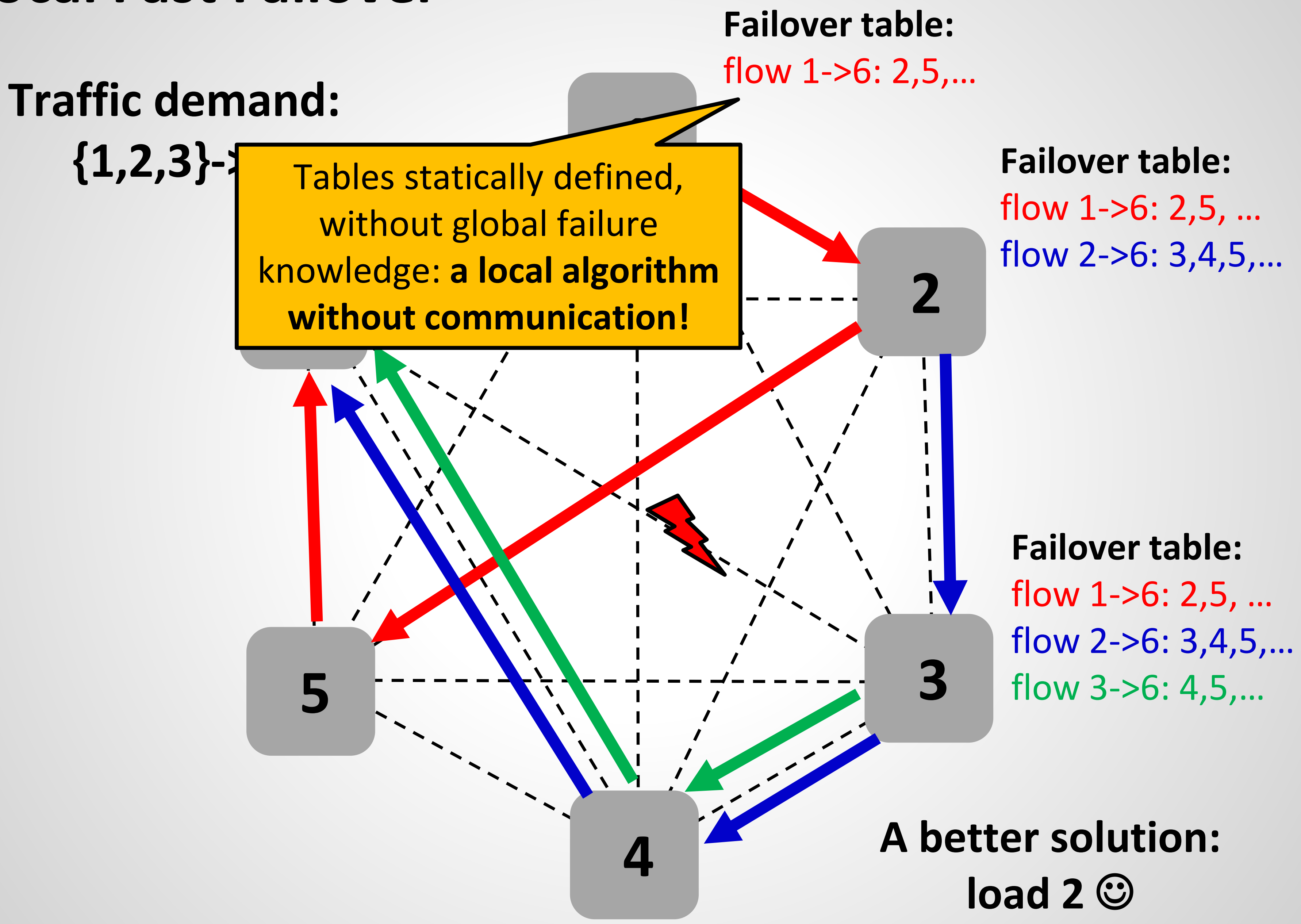
Local Fast Failover

Traffic demand:
{1,2,3}->6



A better solution:
load 2 ☺

Local Fast Failover

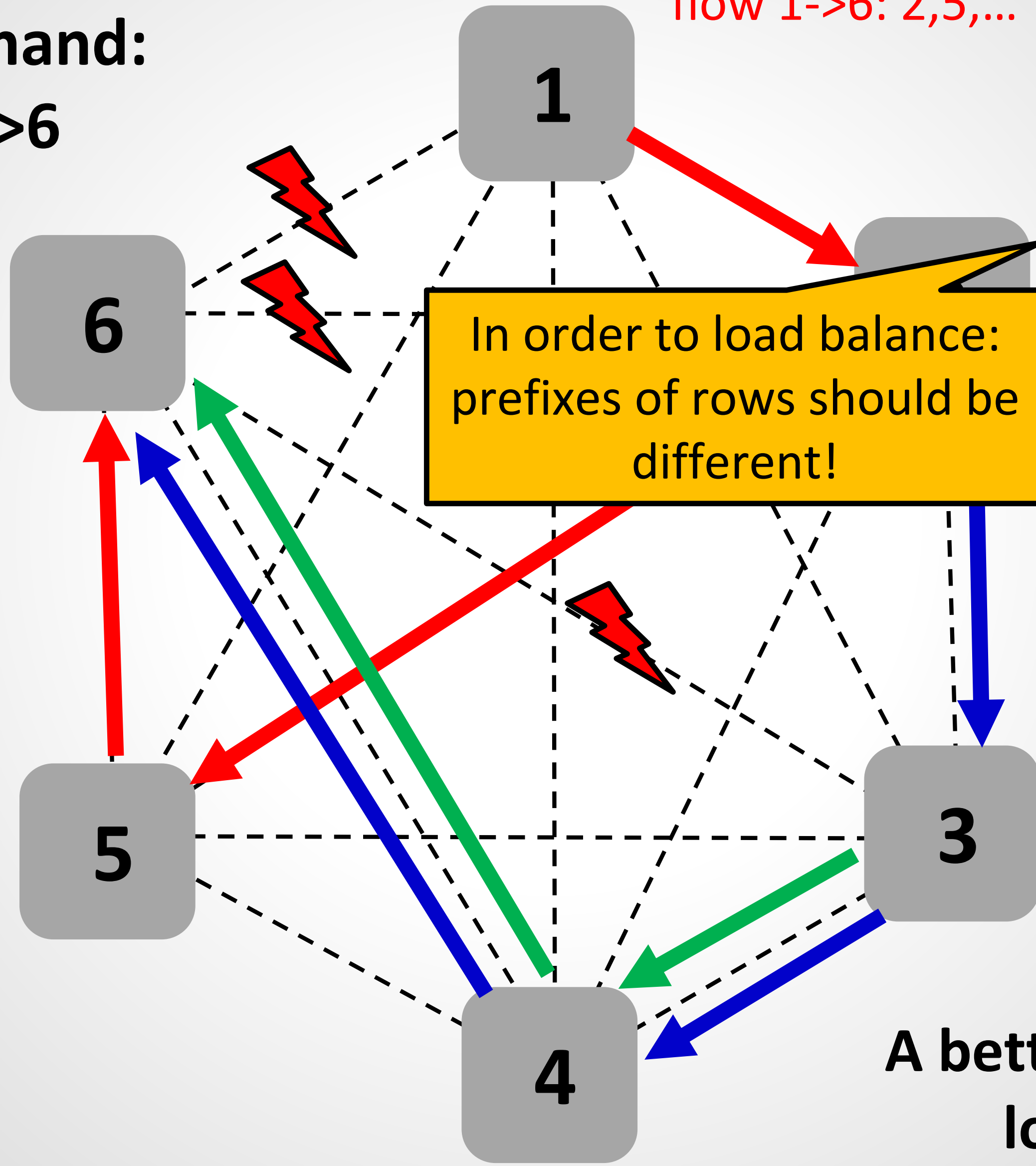


Local Fast Failover

Traffic demand:
{1,2,3}->6

Failover table:
flow 1->6: 2,5,...

Failover table:
flow 1->6: 2,5, ...
flow 2->6: 3,4,5,...



Failover table:
flow 1->6: 2,5, ...
flow 2->6: 3,4,5,...
flow 3->6: 4,5,...

A better solution:
load 2 ☺

Local Fast Failover

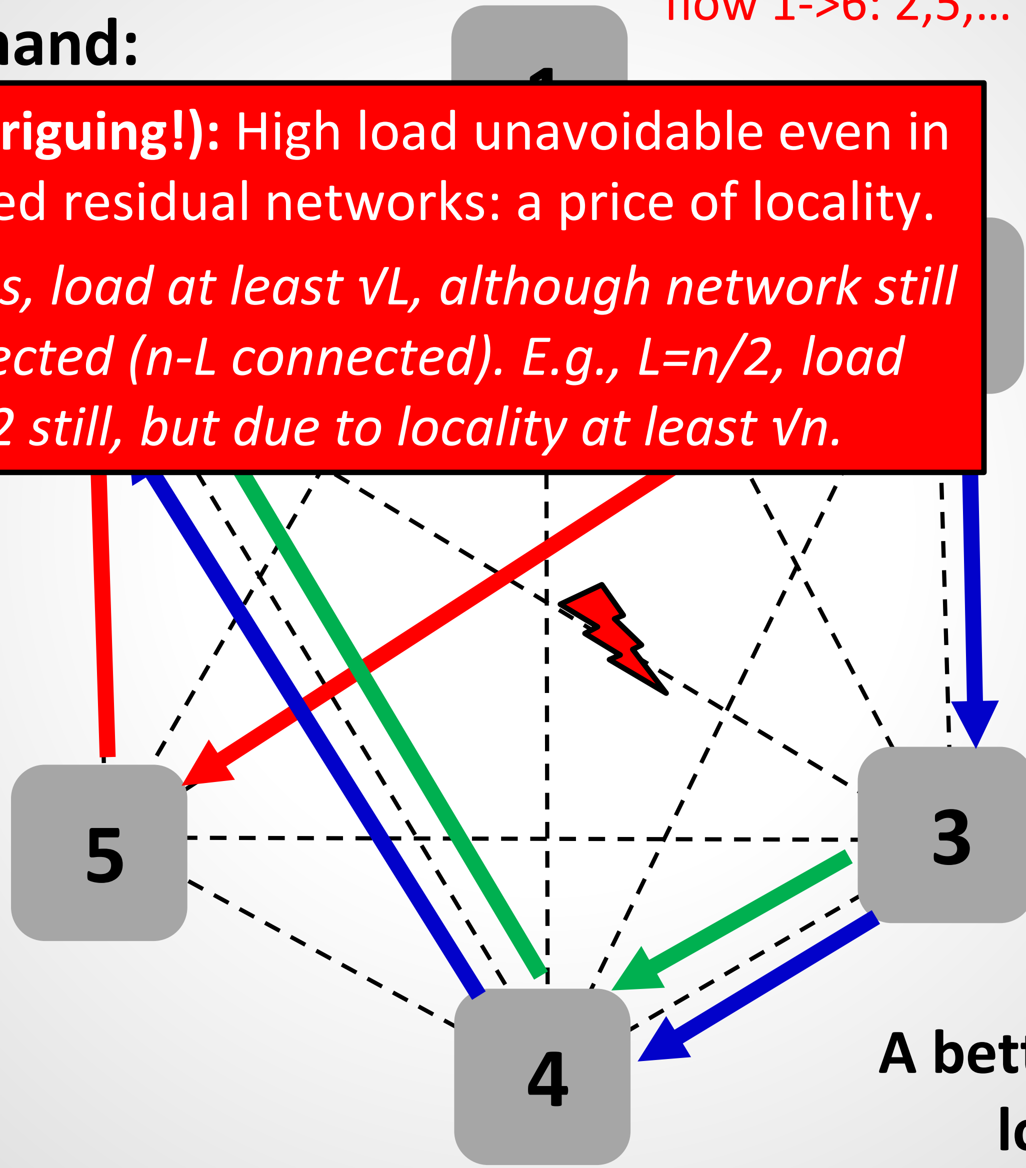
Traffic demand:

Failover table:
flow 1->6: 2,5,...

Bad news (intriguing!): High load unavoidable even in well-connected residual networks: a price of locality.

Given L failures, load at least √L, although network still highly connected (n-L connected). E.g., L=n/2, load could be 2 still, but due to locality at least √n.

Failover table:
flow 1->6: 2,5, ...
flow 2->6: 3,4,5,...



Failover table:
flow 1->6: 2,5, ...
flow 2->6: 3,4,5,...
flow 3->6: 4,5,...

A better solution:
load 2 ☺

Local Fast Failover

Failover table:
flow 1->6: 2,5,...

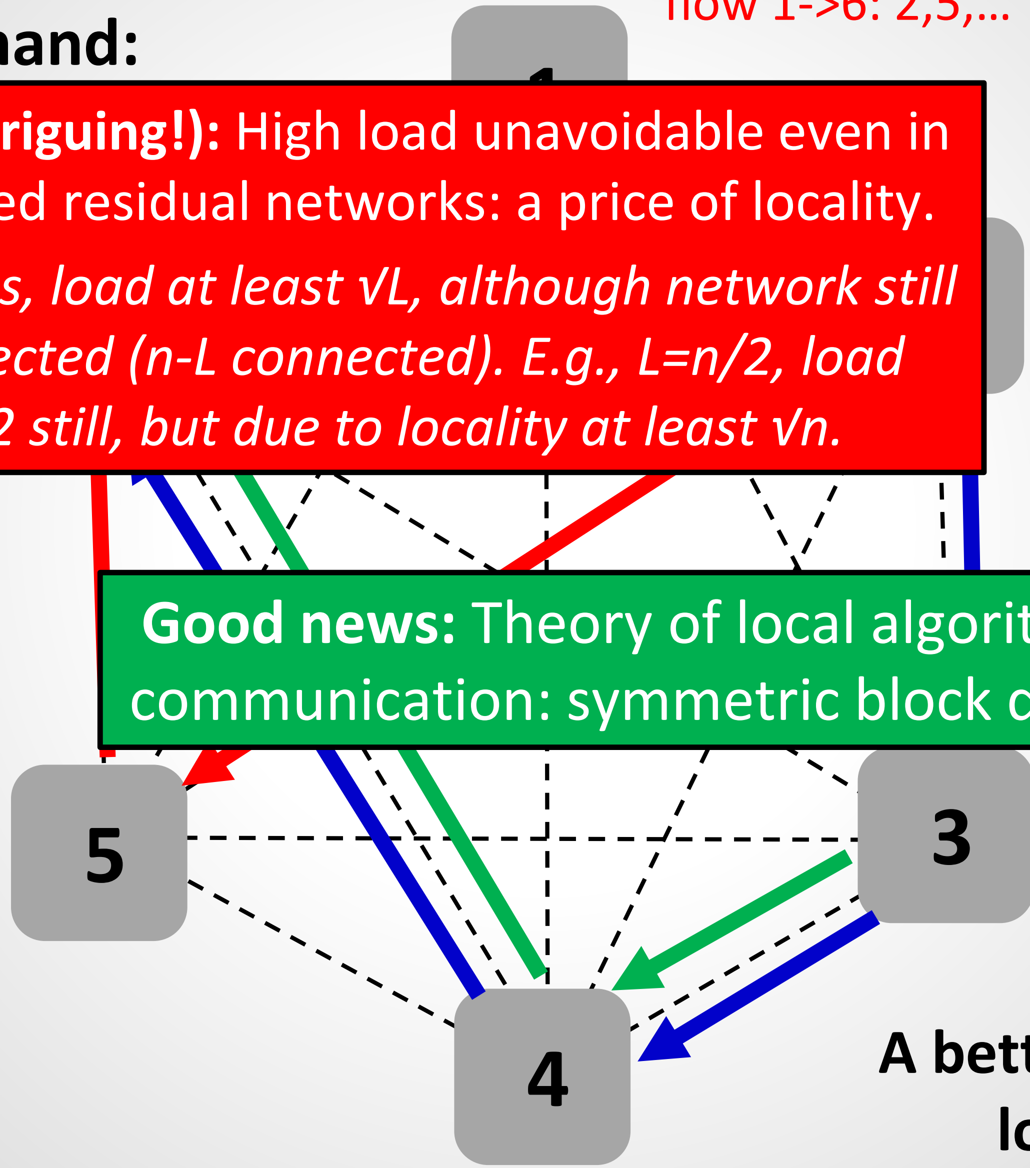
Traffic demand:

Bad news (intriguing!): High load unavoidable even in well-connected residual networks: a price of locality.

Given L failures, load at least √L, although network still highly connected (n-L connected). E.g., L=n/2, load could be 2 still, but due to locality at least √n.

Failover table:
flow 1->6: 2,5, ...
flow 2->6: 3,4,5,...

Good news: Theory of local algorithms without communication: symmetric block design theory.



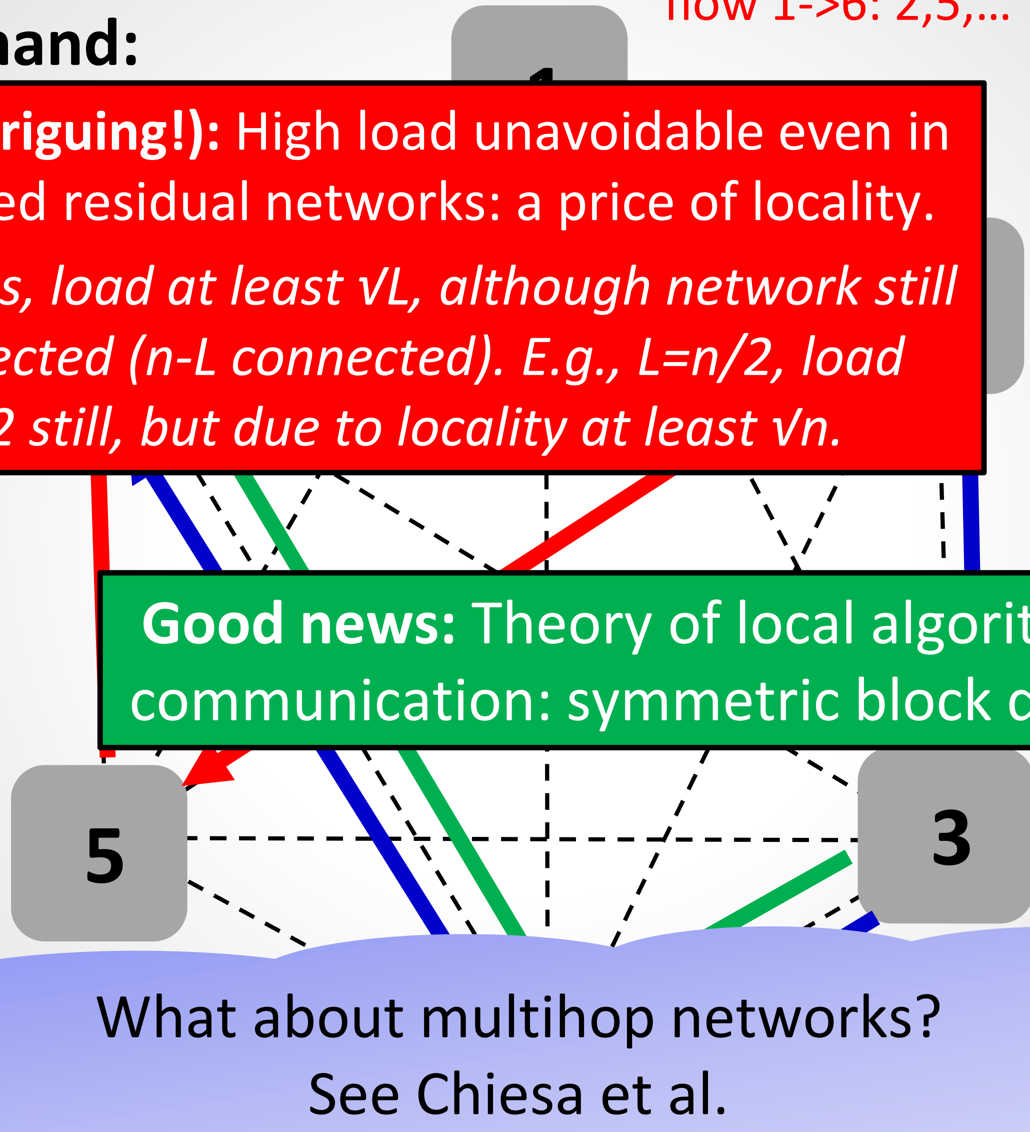
flow 2->6: 3,4,5,...
flow 3->6: 4,5,...

A better solution:
load 2 ☺

Local Fast Failover

Failover table:
flow 1->6: 2,5,...

Traffic demand:



Bad news (intriguing!): High load unavoidable even in well-connected residual networks: a price of locality.

Given L failures, load at least √L, although network still highly connected (n-L connected). E.g., L=n/2, load could be 2 still, but due to locality at least √n.

Failover table:
flow 1->6: 2,5, ...
flow 2->6: 3,4,5,...

Good news: Theory of local algorithms without communication: symmetric block design theory.

flow 2->6: 3,4,5,...
flow 3->6: 4,5,...

What about multihop networks?
See Chiesa et al.

What About Failover
in Multi-Hop Networks?

Solution: Use Arborescences (Chiesa et al.)

- Let us focus on resiliency only and ignore load

- Assume:
  - single destination *d* and *k*-connected network *G*
  - *G* decomposed into *k* *d*-rooted arc-disjoint spanning arborescences

always exist in k-connected graphs (efficient)

Basic principle:

- Route along fixed arborescence towards the destination *d*
- If packet hits a failed edge at vertex *v*, reroute along a different arborescence

Solution: Use Arborescences (Chiesa et al.)

- Let us focus on resiliency only and ignore load

- Assume:
  - single destination *d* and *k*-connected network *G*
  - *G* decomposed into *k d*-rooted arc-disjoint spanning arborescences

always exist in k-connected graphs (efficient)

Basic principle:

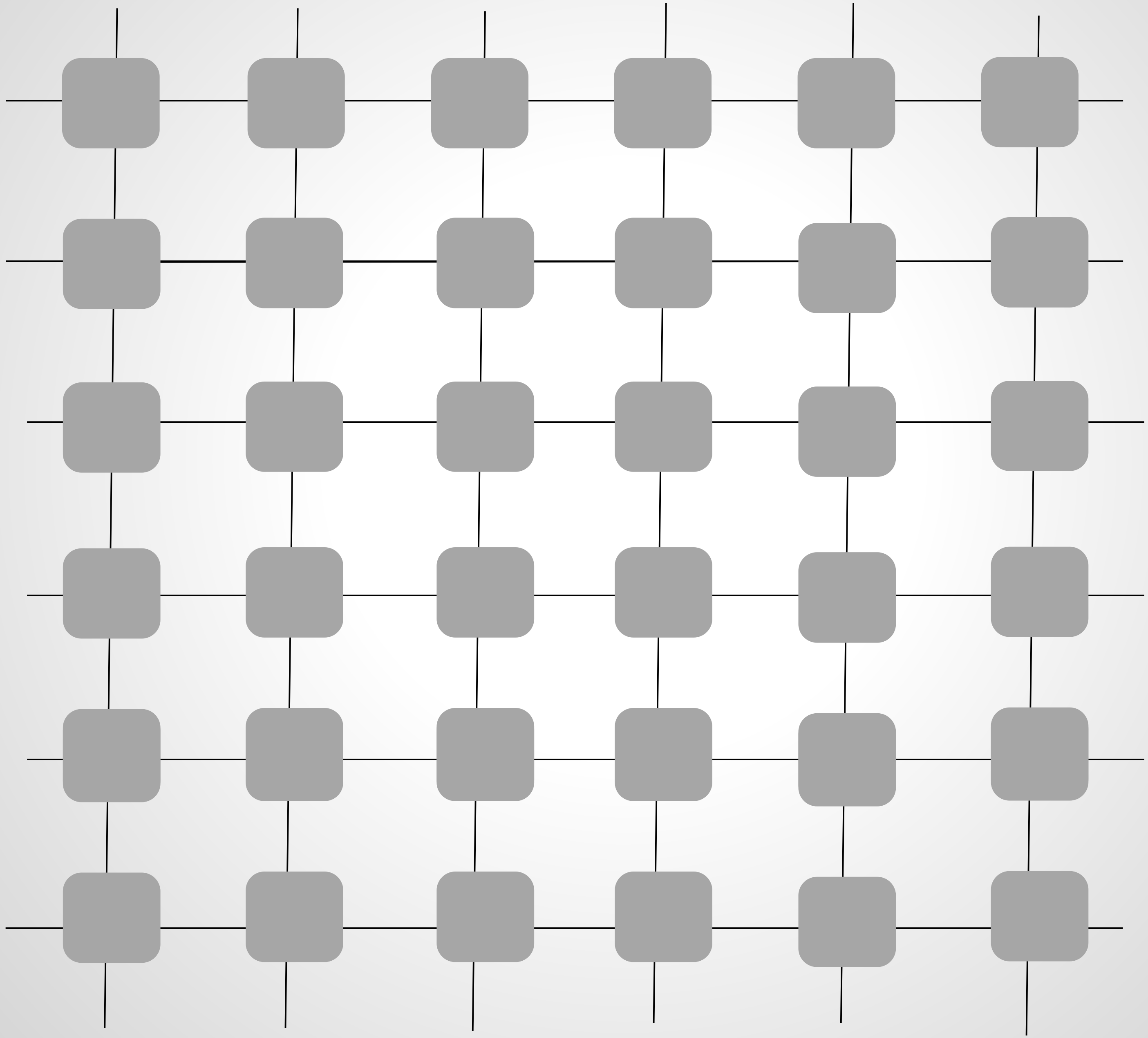
- Route along fixed arborescence towards the destination *d*
- If packet hits a failed edge at vertex *v*, reroute along a different arborescence

The crux: To which one? Random? Influences resilience.

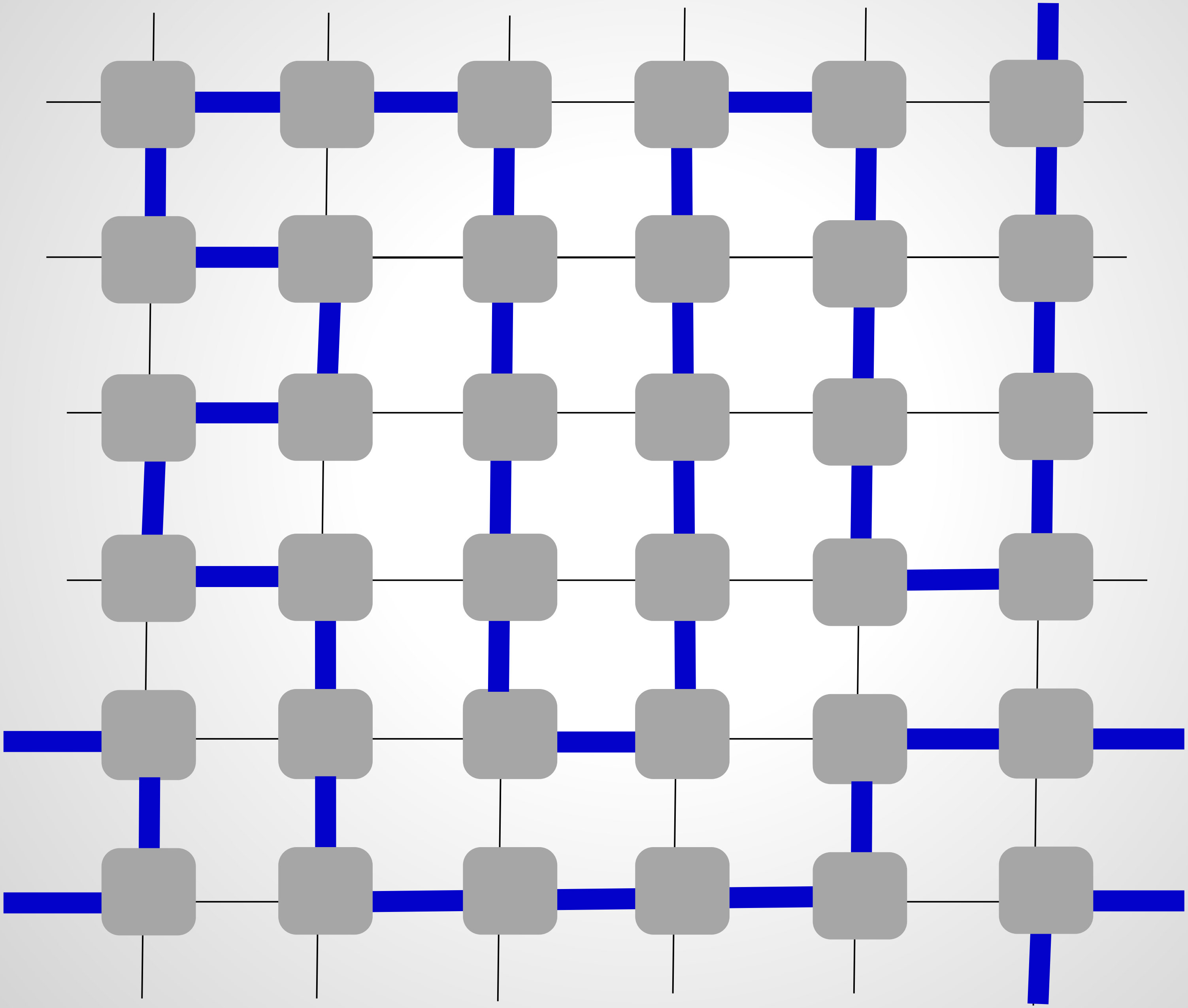
Simple Example: Hamilton Cycle

Chiesa et al.: if *k*-connected graph has *k* arc disjoint Hamilton Cycles, *k*-1 resilient routing can be constructed!

Example: 3-Resilient Routing Function for 2d-Torus

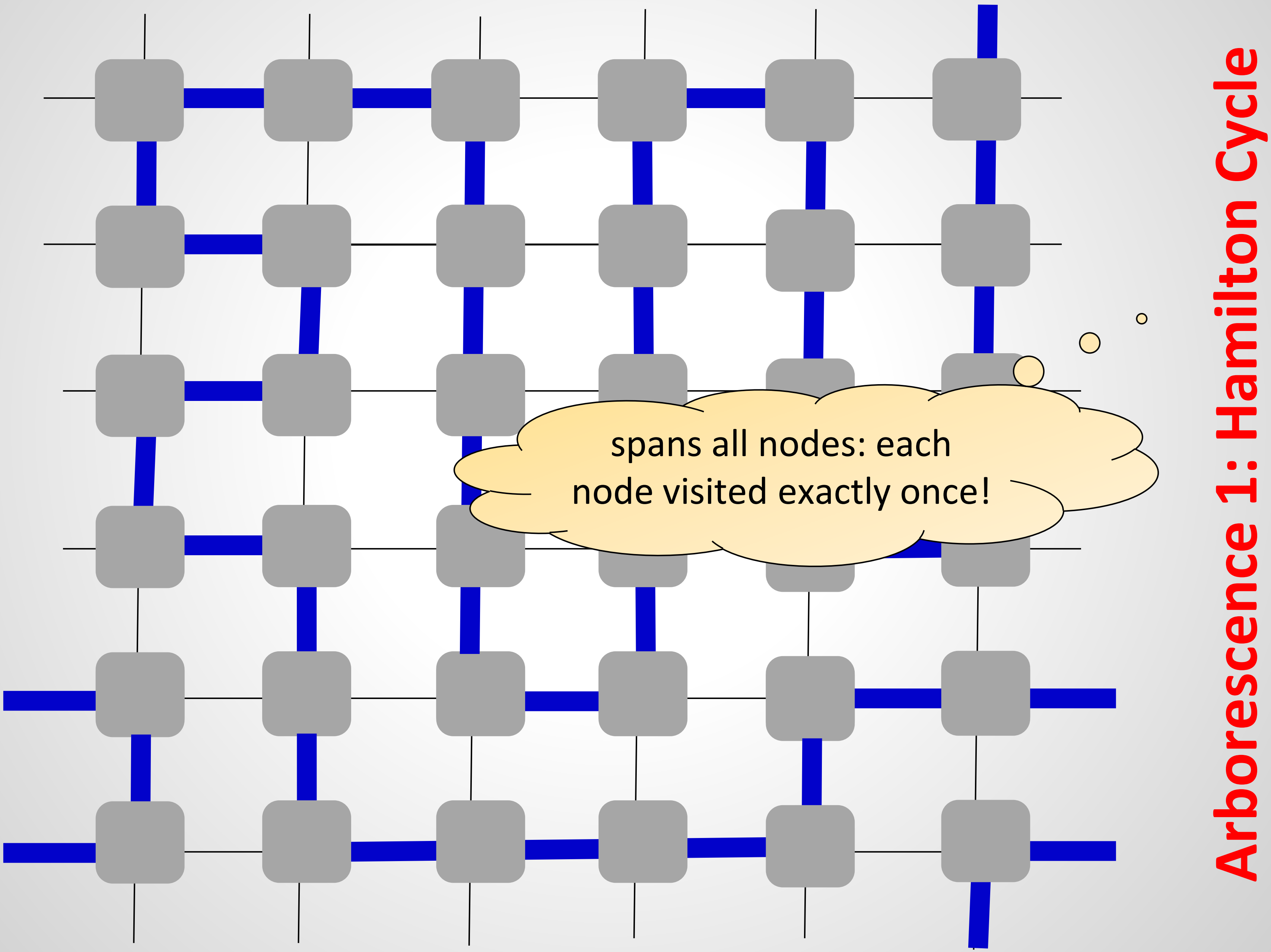


Example: 3-Resilient Routing Function for 2d-Torus

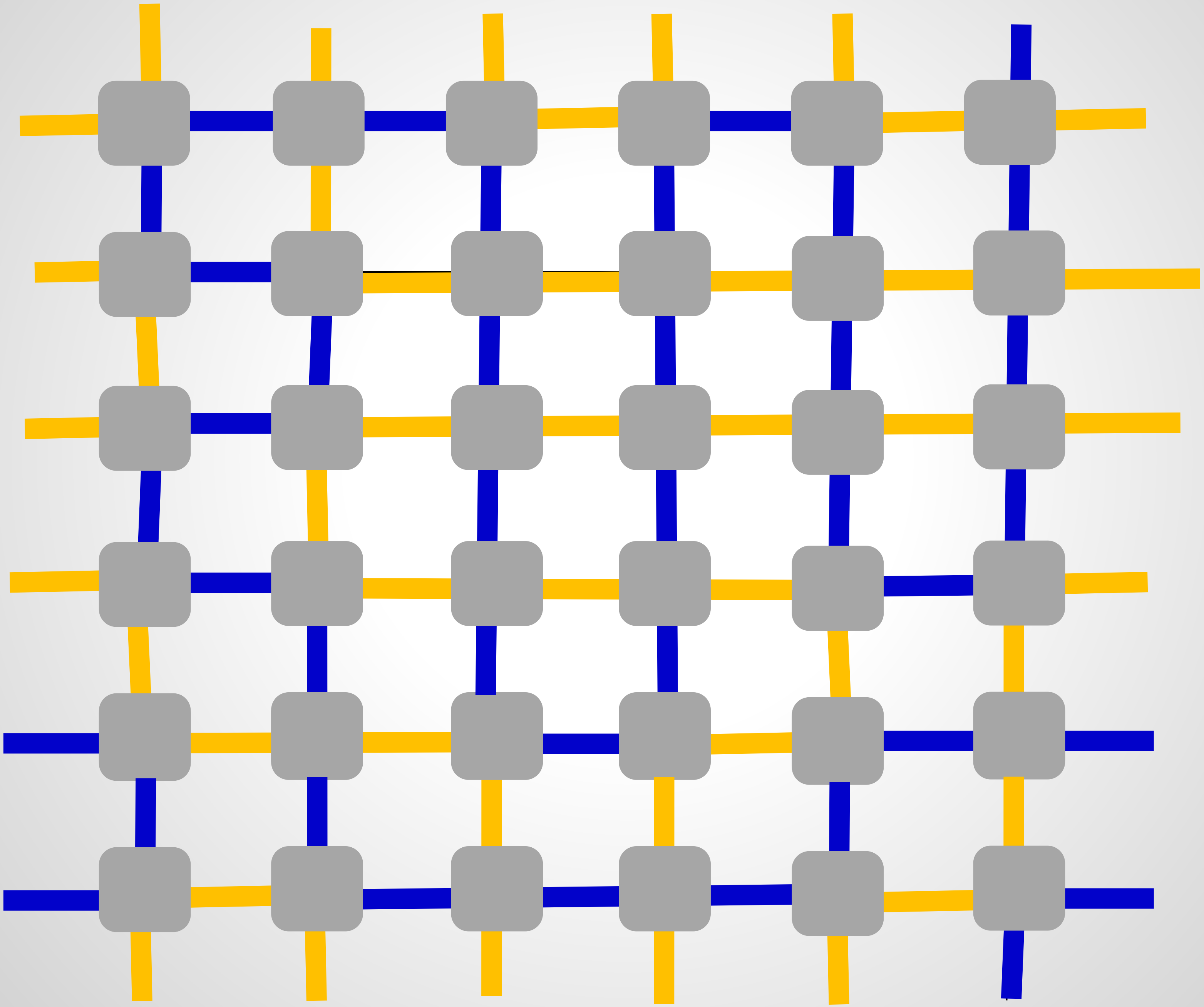


Arborescence 1: Hamilton Cycle

Example: 3-Resilient Routing Function for 2d-Torus

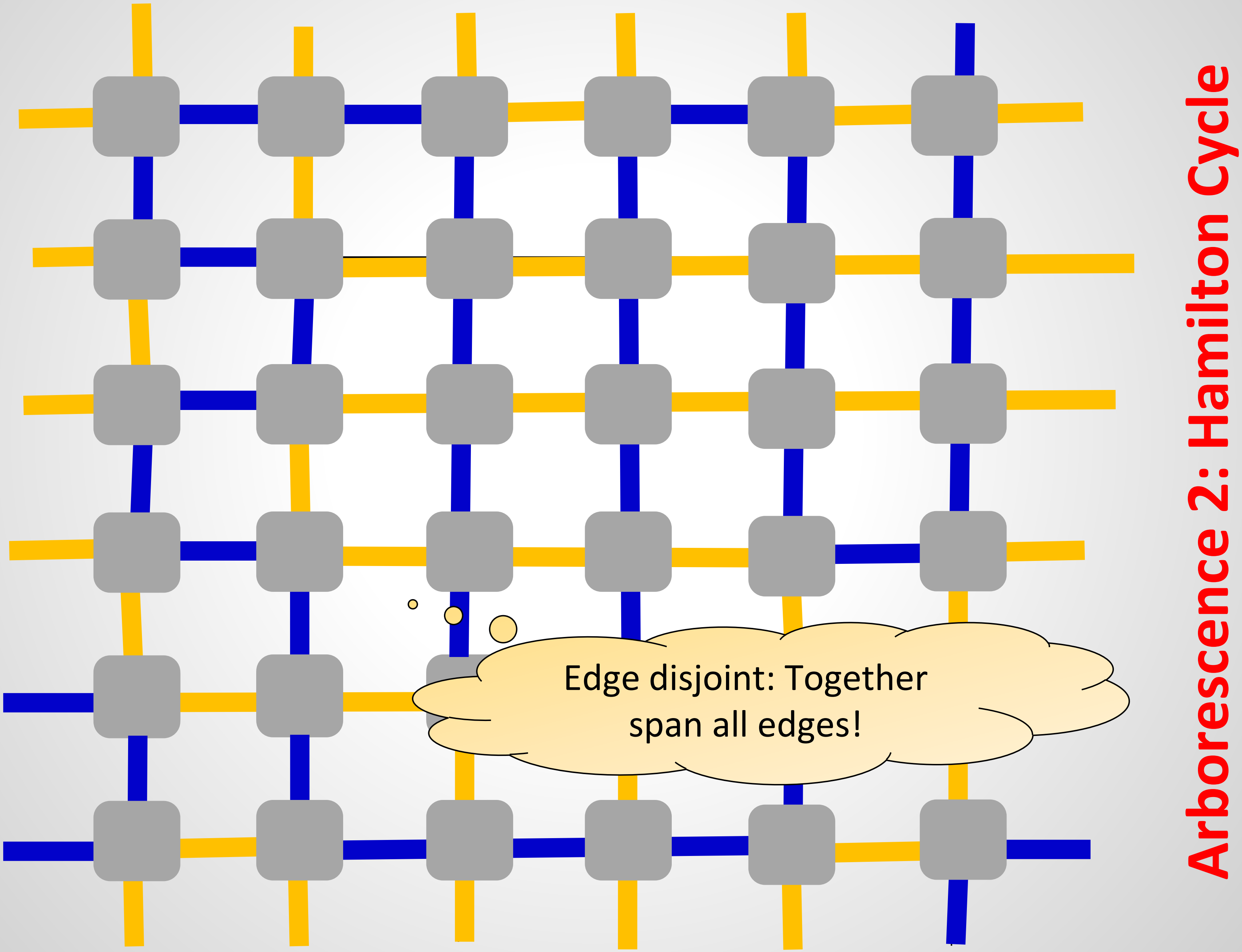


Example: 3-Resilient Routing Function for 2d-Torus

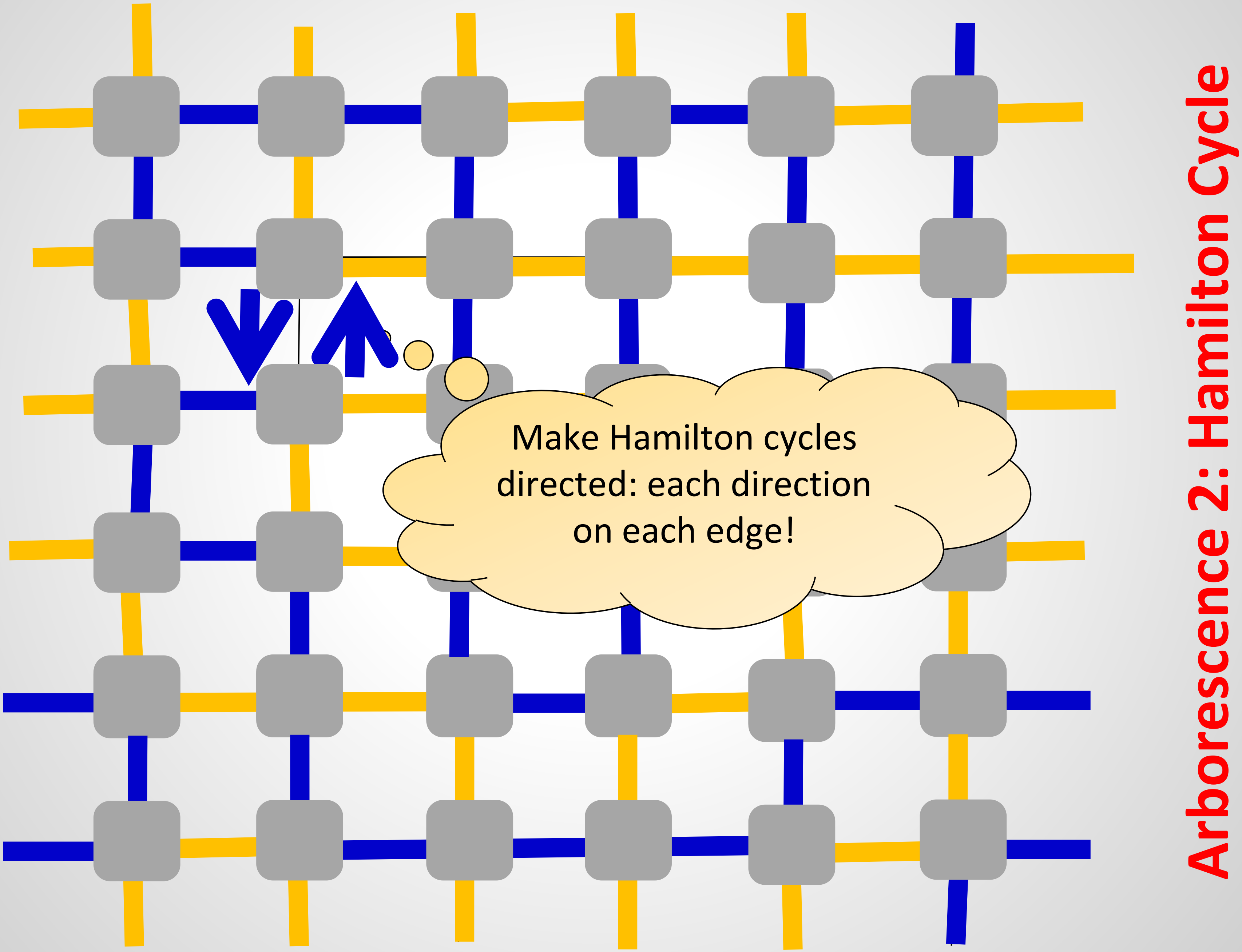


Arborescence 2: Hamilton Cycle

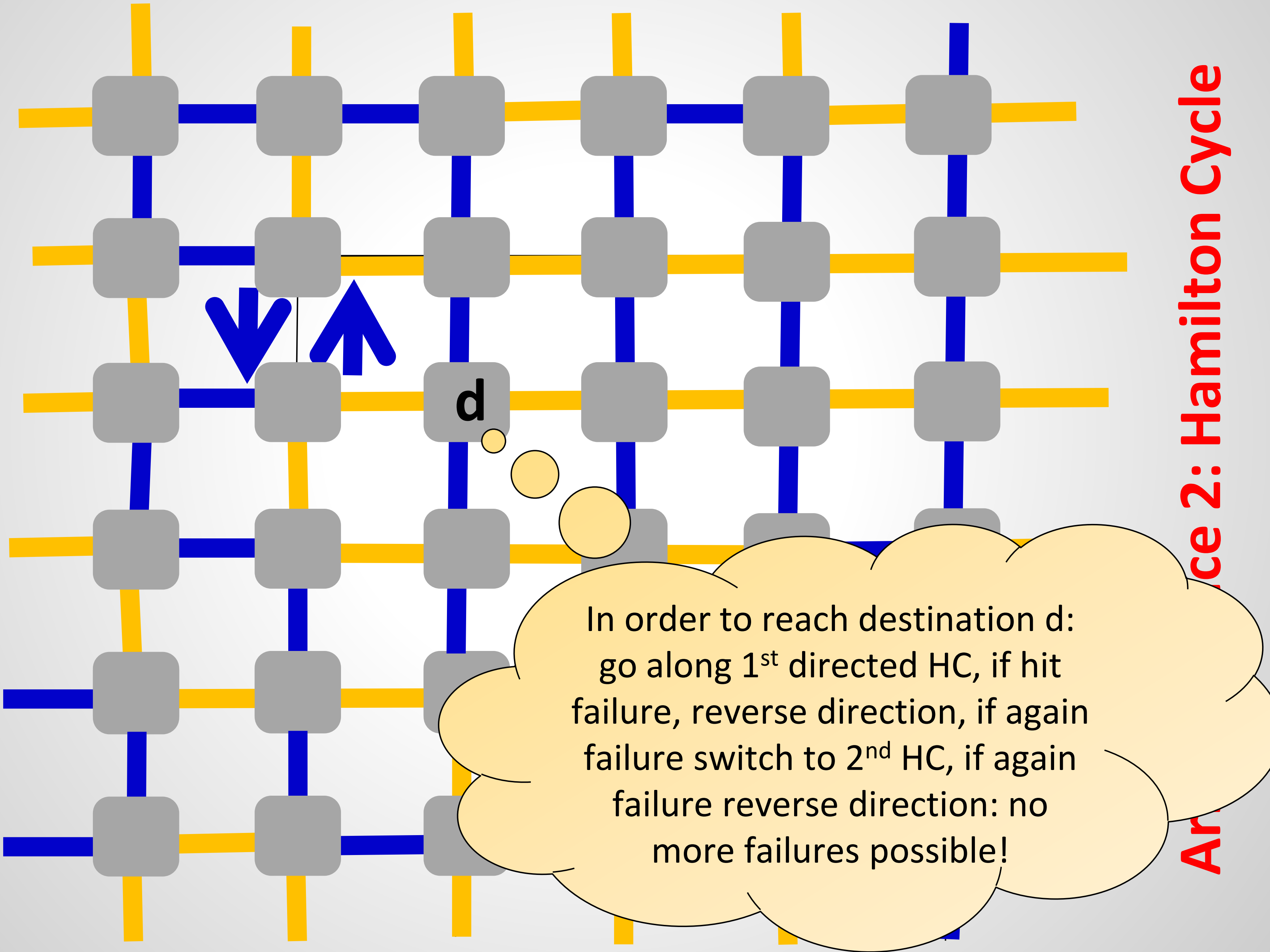
Example: 3-Resilient Routing Function for 2d-Torus



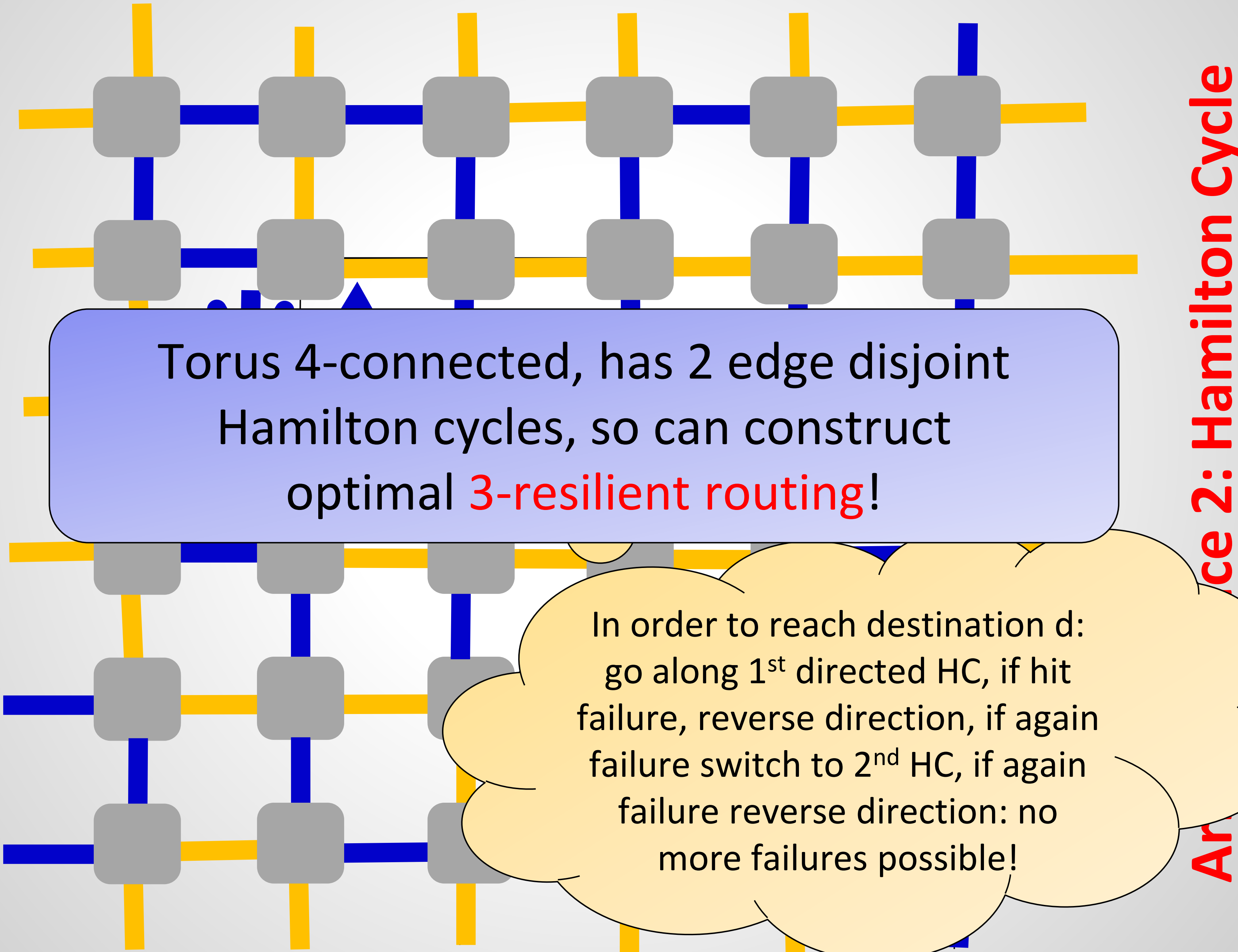
Example: 3-Resilient Routing Function for 2d-Torus



Example: 3-Resilient Routing Function for 2d-Torus



Example: 3-Resilient Routing Function for 2d-Torus



Torus 4-connected, has 2 edge disjoint Hamilton cycles, so can construct optimal 3-resilient routing!

In order to reach destination d: go along 1st directed HC, if hit failure, reverse direction, if again failure switch to 2nd HC, if again failure reverse direction: no more failures possible!

Many Open Problems

For example:

- Optimal robustness: given a *k*-connected graph, can we always find a failover scheme which is *k*-resilient?

- If not, what is the «local failover robustness» of a given graph?

Further Reading

Load-Optimal Local Fast Rerouting for Dependable Networks
Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan.
47th IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Denver, Colorado, USA, June 2017.

How (Not) to Shoot in Your Foot with SDN Local Fast Failover: A Load-Connectivity Tradeoff
Michael Borokhovich and Stefan Schmid.
17th International Conference on Principles of Distributed Systems (**OPODIS**), Nice, France, Springer LNCS, December 2013.

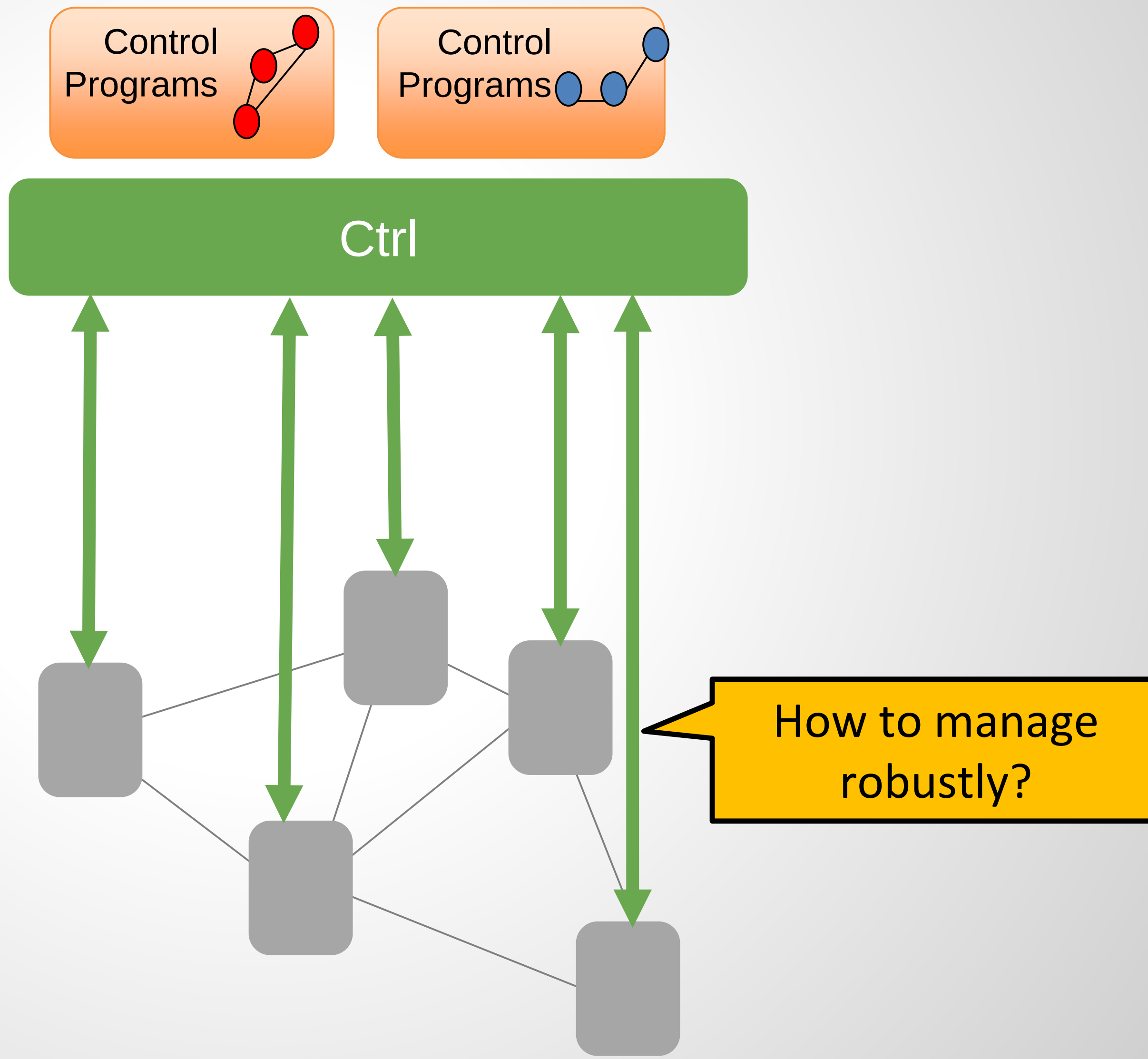
Exploring the Limits of Static Resilient Routing
Marco Chiesa et al. **arXiv** Report, **ICALP** 2016, **INFOCOM** 2016.

Algorithmic Problems in SDNs

Applications and Control Plane

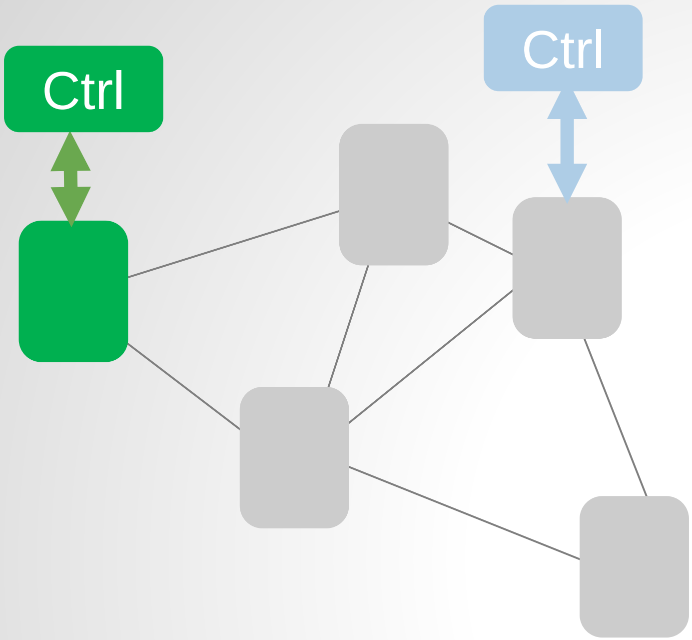
... and regarding decoupling / inter-connect!

Data Plane

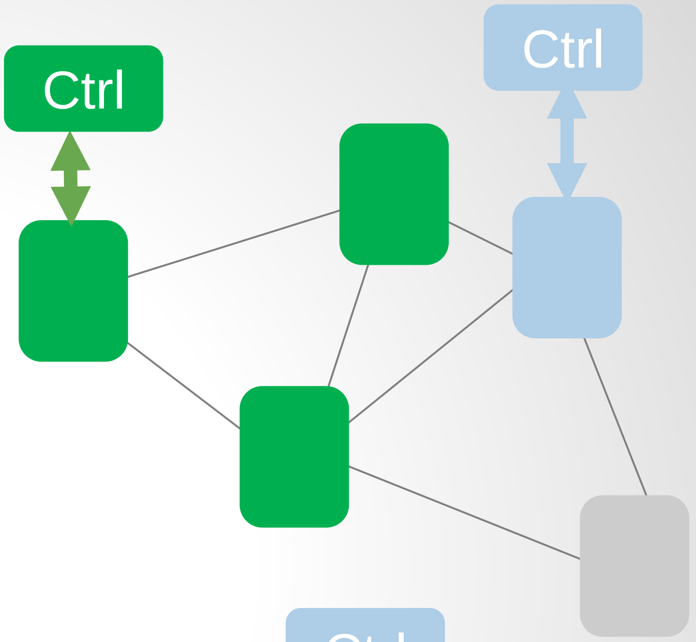


A Self-Stabilization Problem!

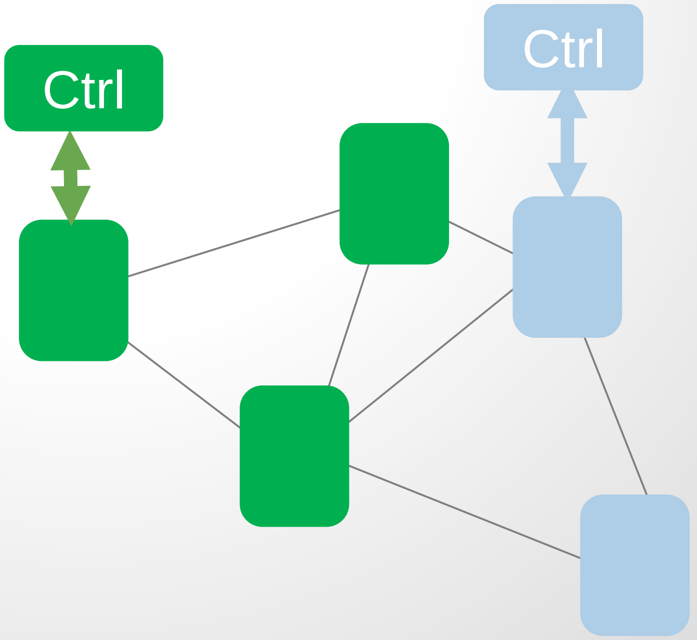
1.



2.



3.



Further Reading

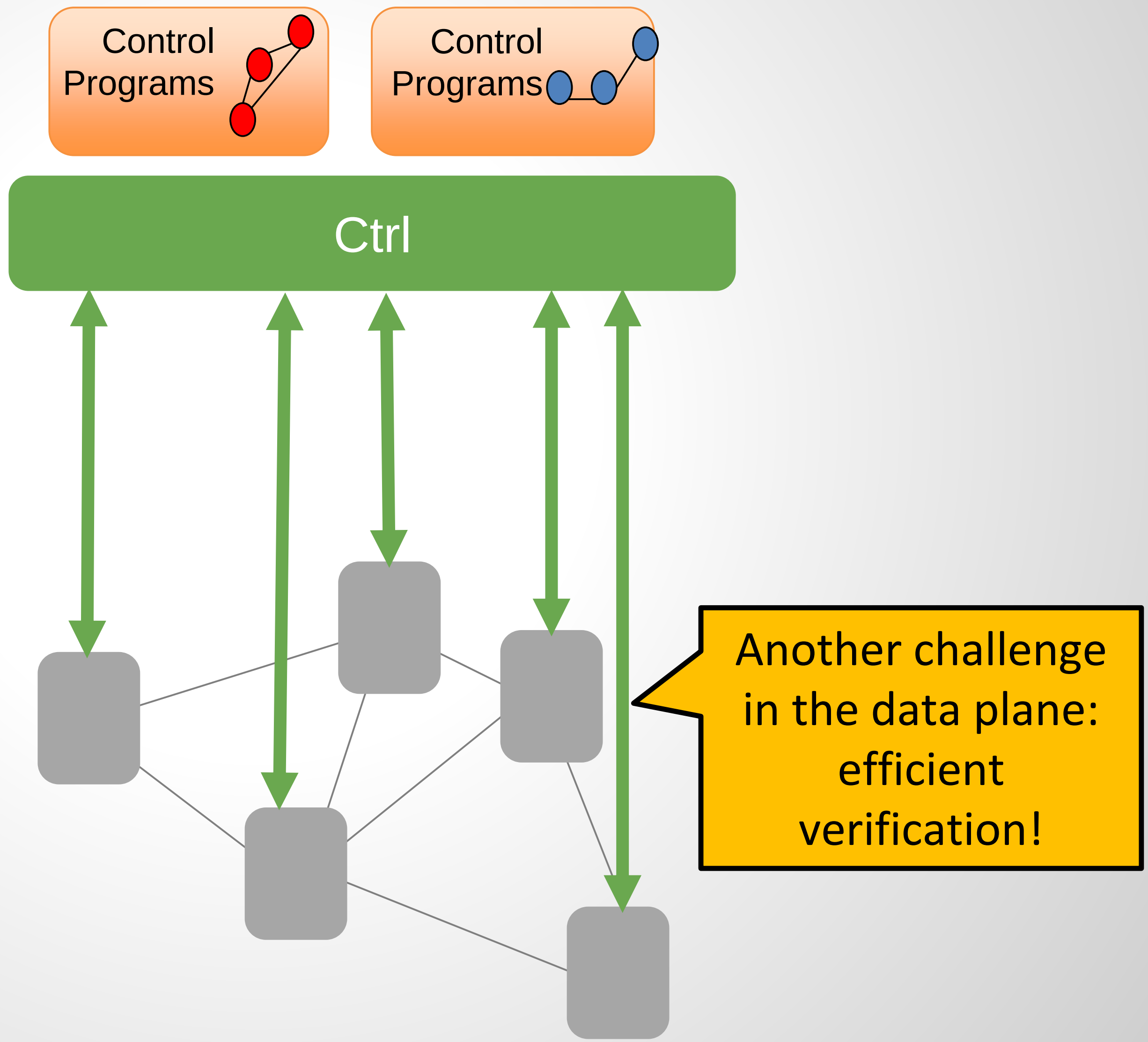
A Self-Organizing Distributed and In-Band SDN Control Plane (Poster Paper)
Marco Canini, Iosif Salem, Liron Schiff, Elad M. Schiller, and Stefan Schmid.
37th IEEE International Conference on Distributed Computing Systems (**ICDCS**), Atlanta, Georgia, USA, June 2017.

Algorithmic Problems in SDNs

Applications and Control Plane

... and regarding decoupling / inter-connect!

Data Plane

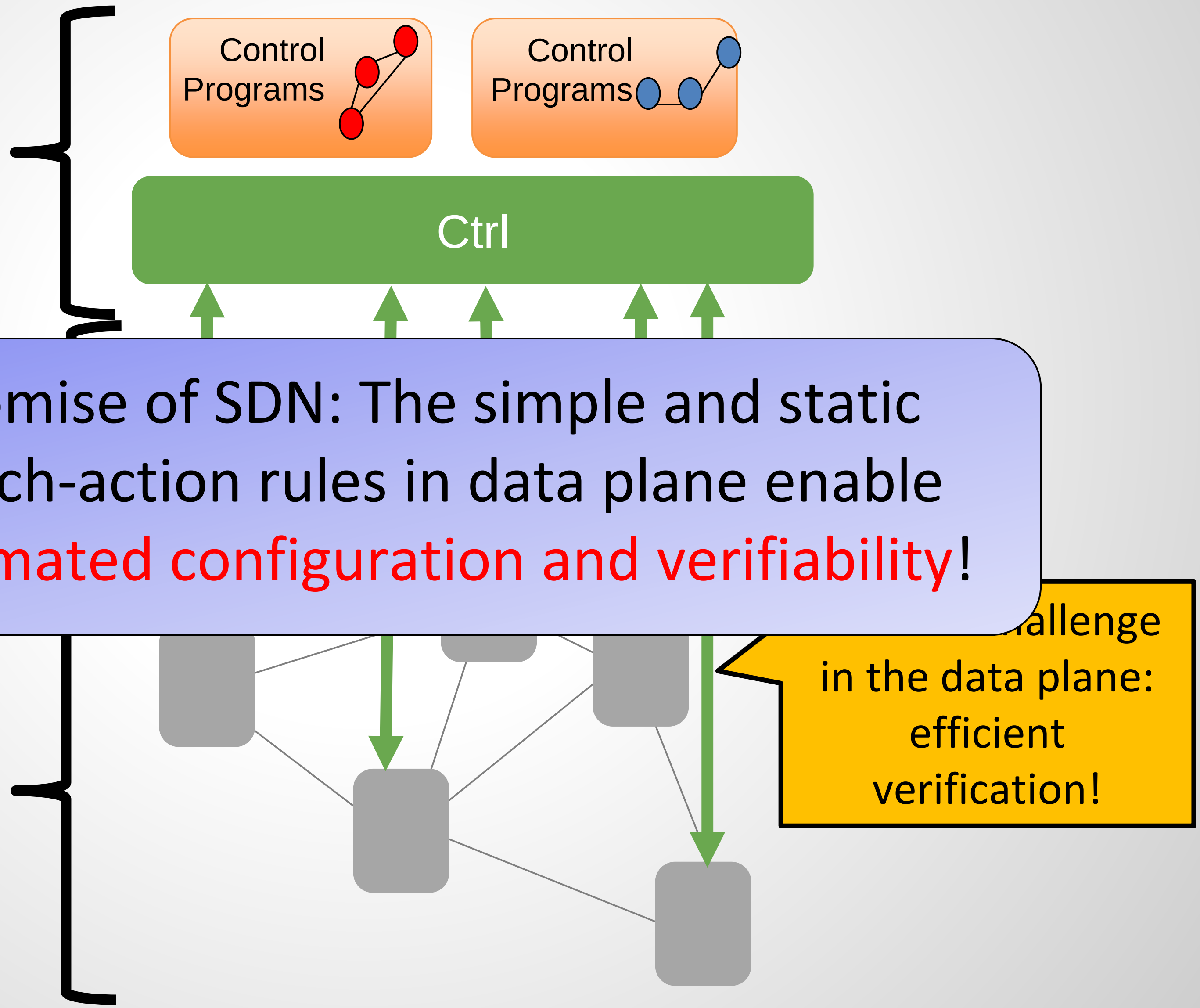


Algorithmic Problems in SDNs

Applications and Control Plane

… and re
decouplin
conn

Data Plane



Promise of SDN: The simple and static match-action rules in data plane enable automated configuration and verifiability!

Questions Operators May Have:

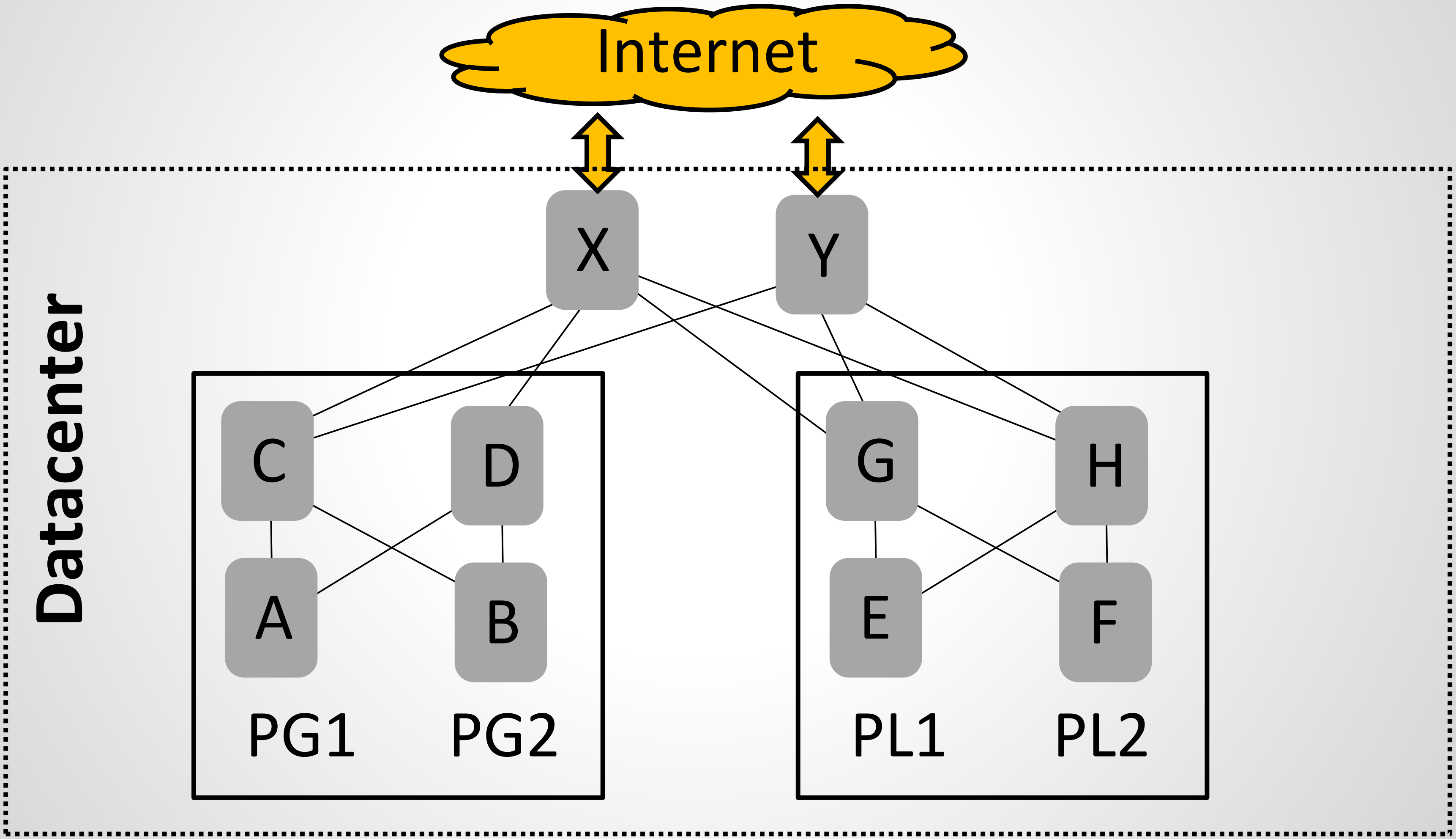
- **Reachability:** «Is it possible / not possible to reach, from ingress port x, egress port y?»
  - To ensure connectivity
  - But also policies: professor network not reachable from student dorms (logical isolation)

- **What-if analysis:** «How can the forwarding behavior look like if there are up to *k* concurrent link failures?»

Policy-compliance under failures is difficult!

What-if Analysis Matters

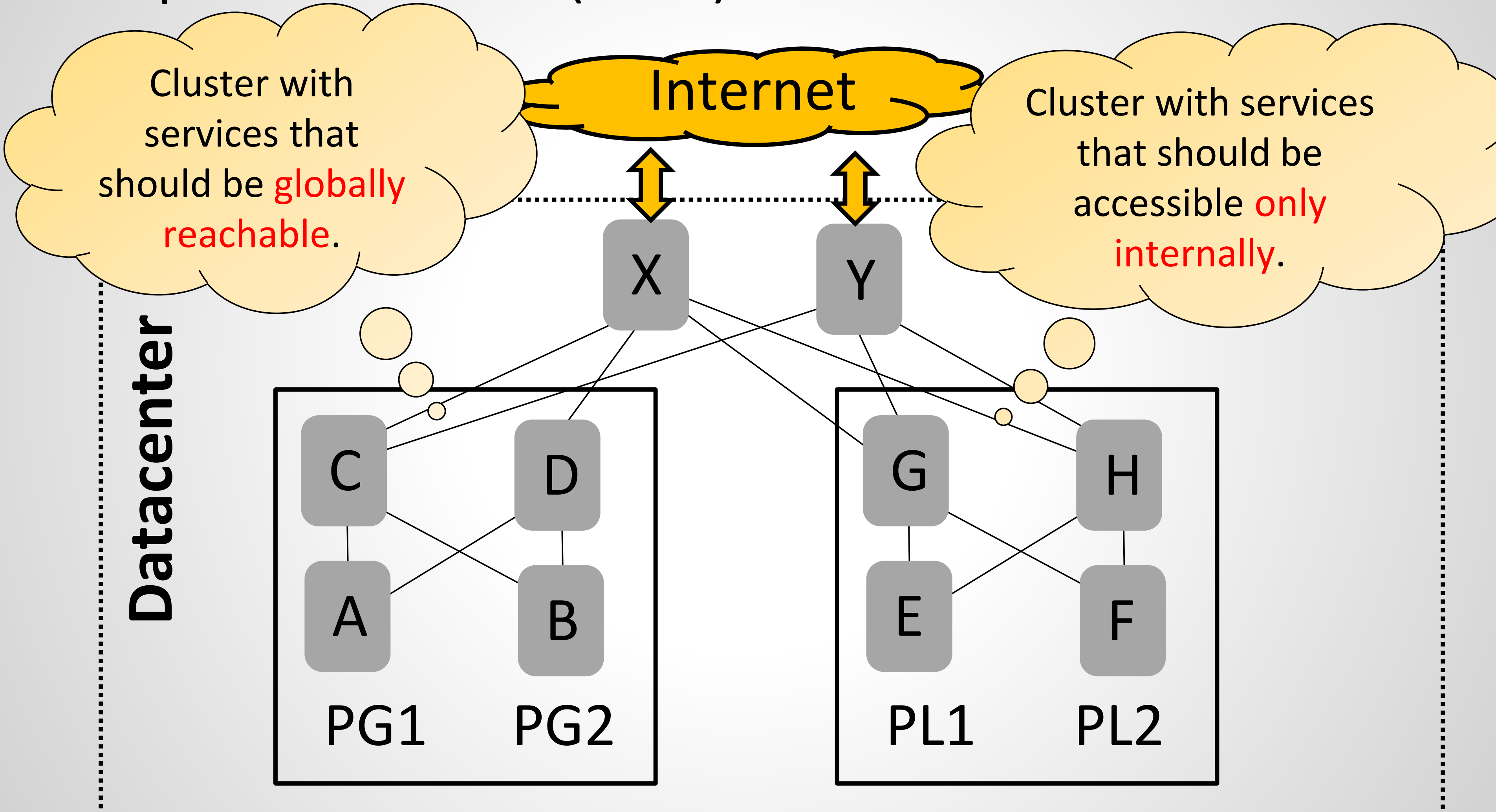
Example: Datacenter (BGP!)*



* *Example taken from* **Beckett et al. (SIGCOMM 2016): Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations.**

What-if Analysis Matters

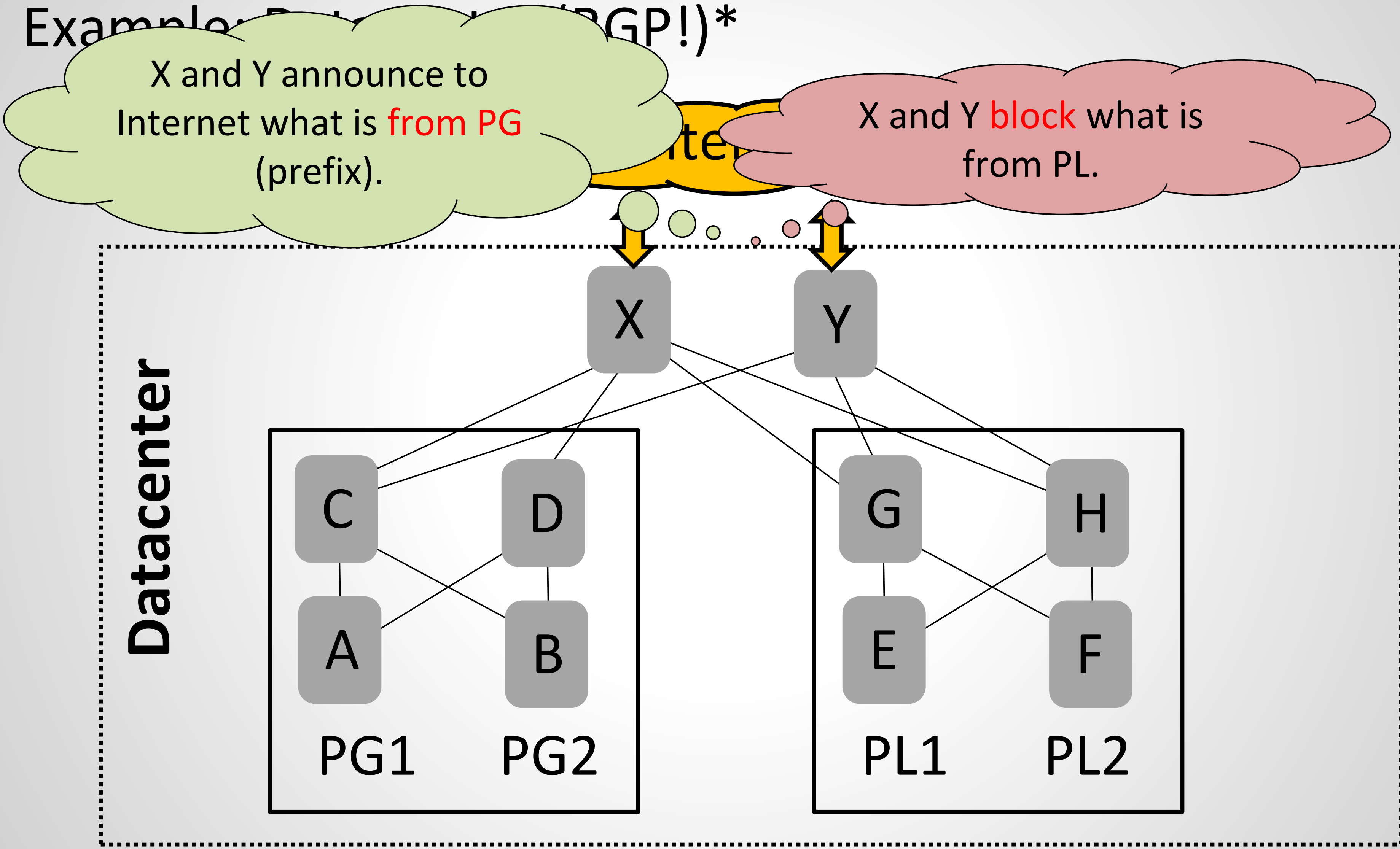
Example: Datacenter (BGP!)*



* *Example taken from* **Beckett et al. (SIGCOMM 2016): Don’t Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations.**

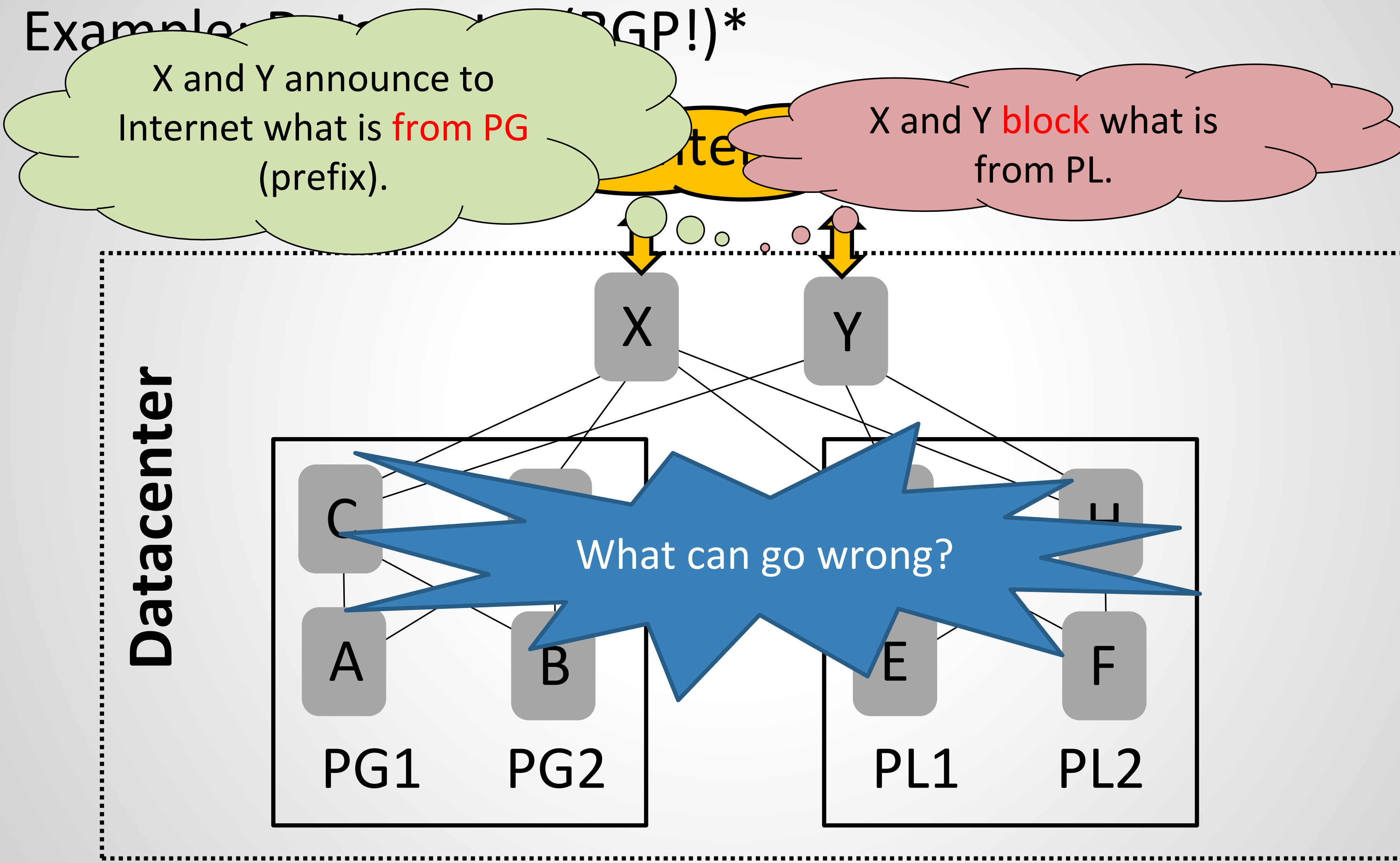
What-if Analysis Matters

Example: Data… (BGP!)*



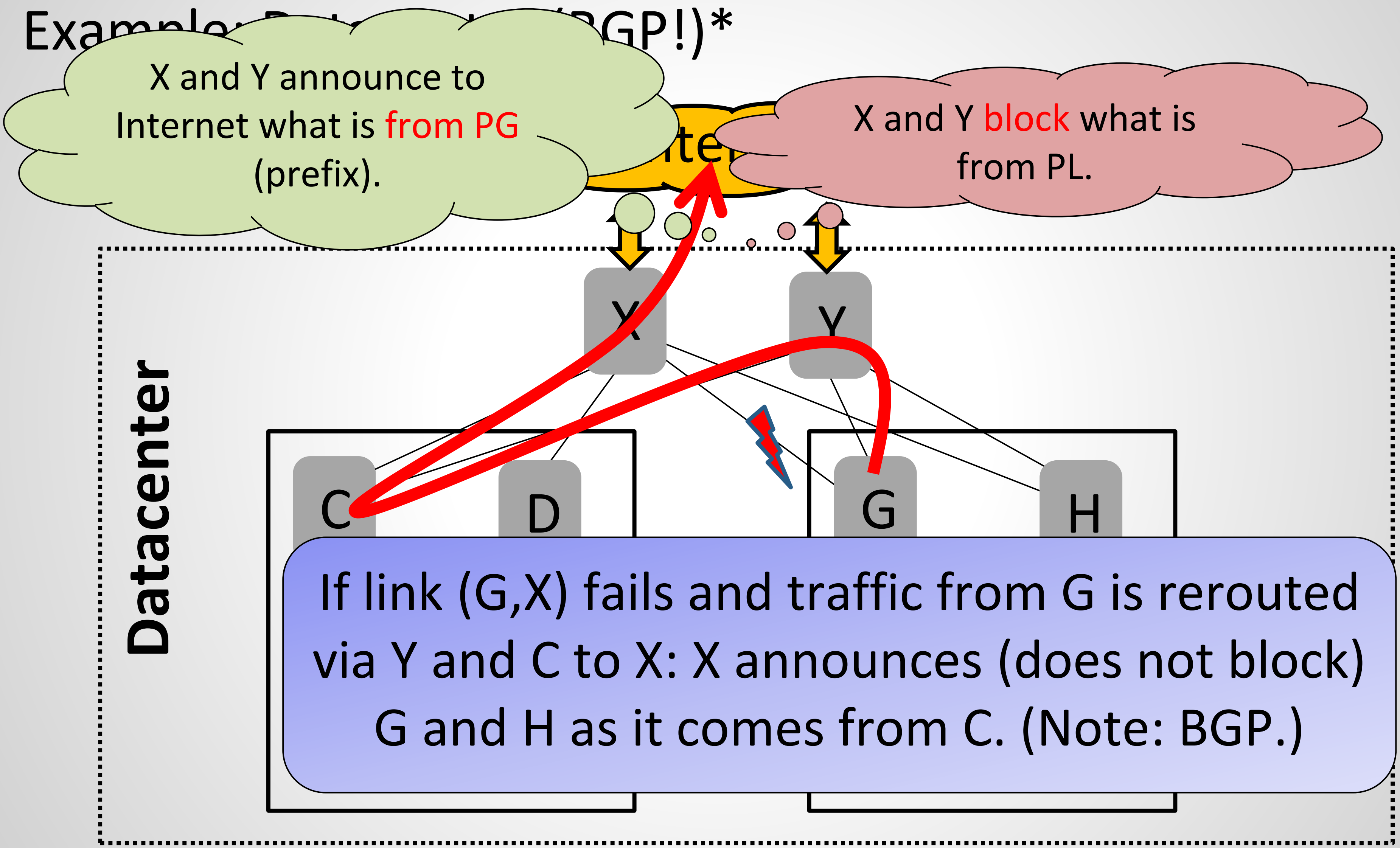
* *Example taken from* **Beckett et al. (SIGCOMM 2016): Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations.**

What-if Analysis Matters



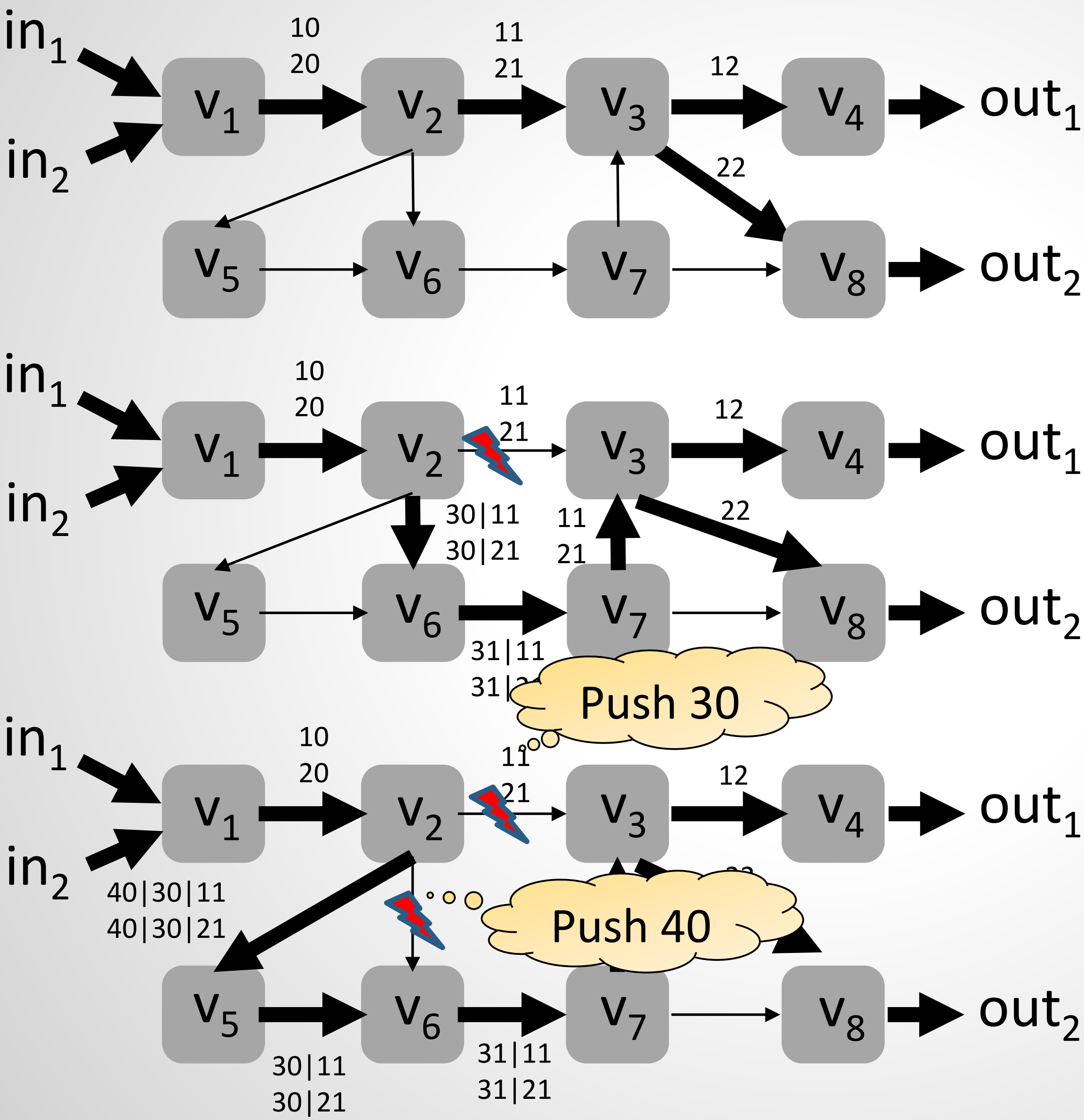
*** *Example taken from* Beckett et al. (SIGCOMM 2016): Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations.**

What-if Analysis Matters



** Example taken from* **Beckett et al. (SIGCOMM 2016): Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations.**

Multiple Link Failures: Push Recursively!



Original Routing

One failure: push 30: route around (v_2,v_3)

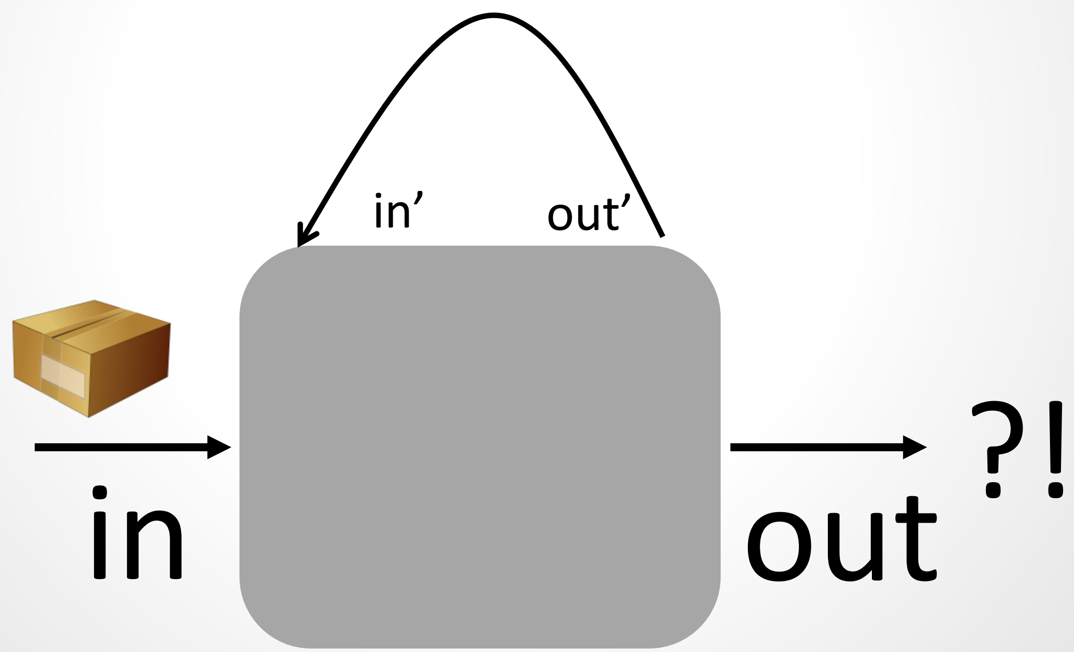
Two failures: first push 30: route around (v_2,v_3)

Recursively push 40: route around (v_2,v_6)

Tractability of Verification

Even without failures: reachability test is undecidable in SDN!

Proof: Can emulate a Turing machine.

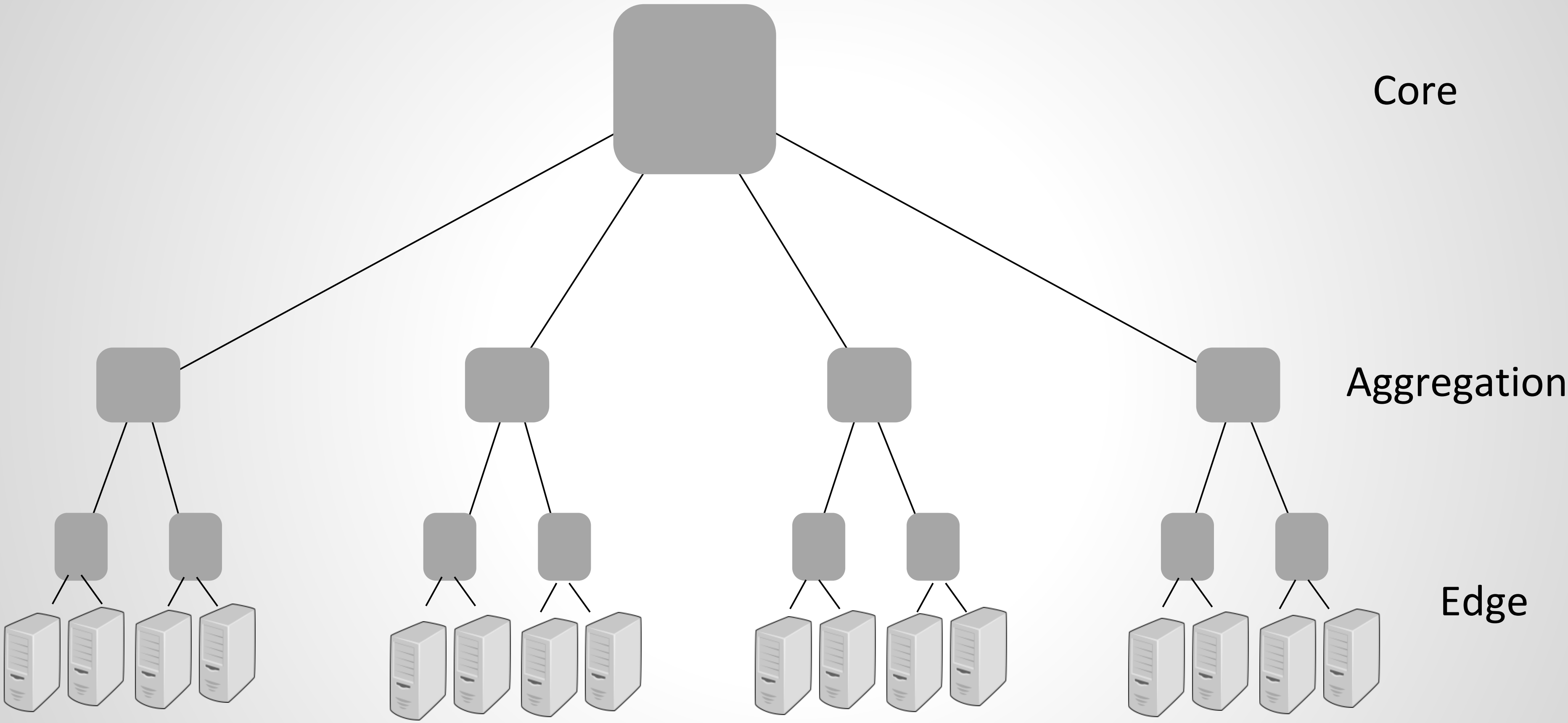


Further Reading

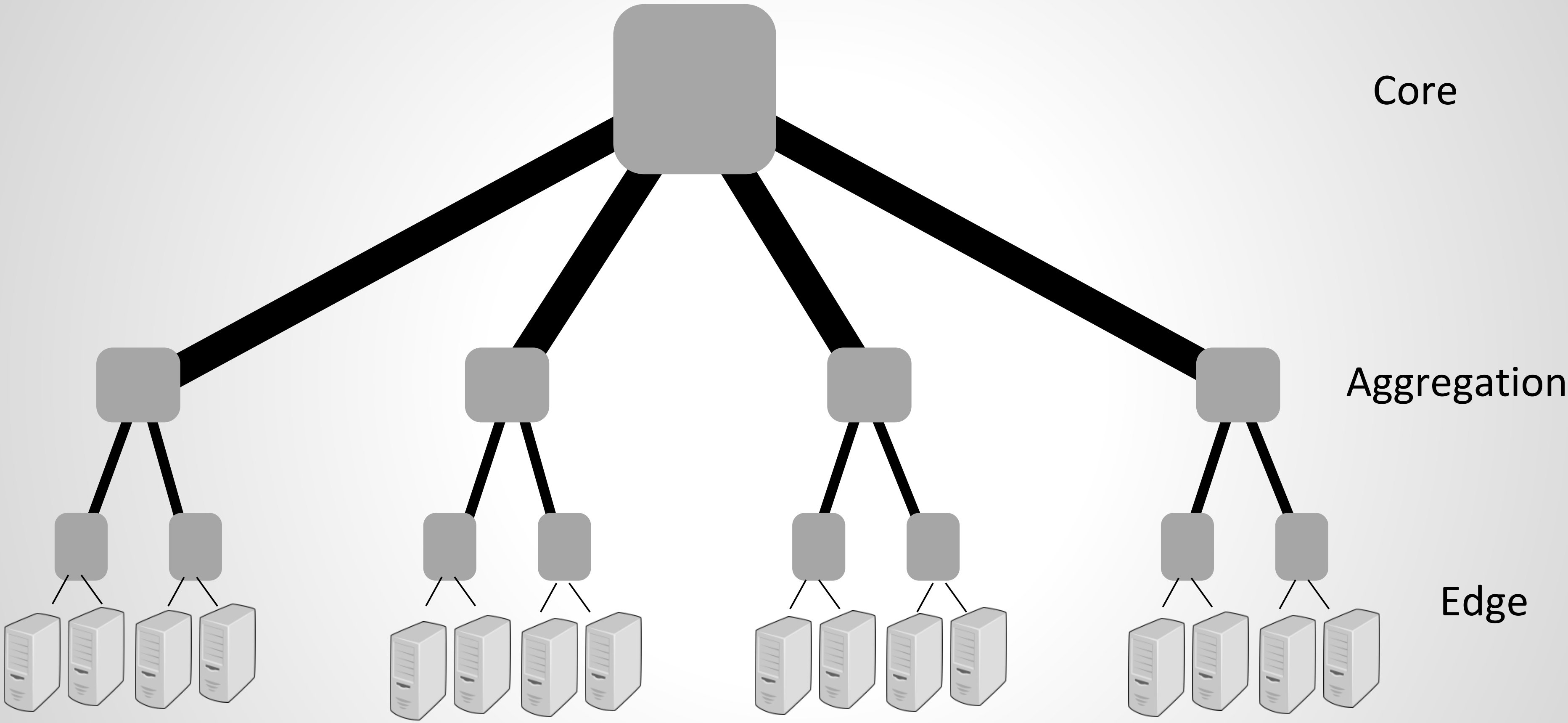
WNetKAT: A Weighted SDN Programming and Verification Language
Kim G. Larsen, Stefan Schmid, and Bingtian Xue.
20th International Conference on Principles of Distributed Systems (**OPODIS**), Madrid, Spain, December 2016.

Another Emerging Flexibility:
Reconfigurable Topologies

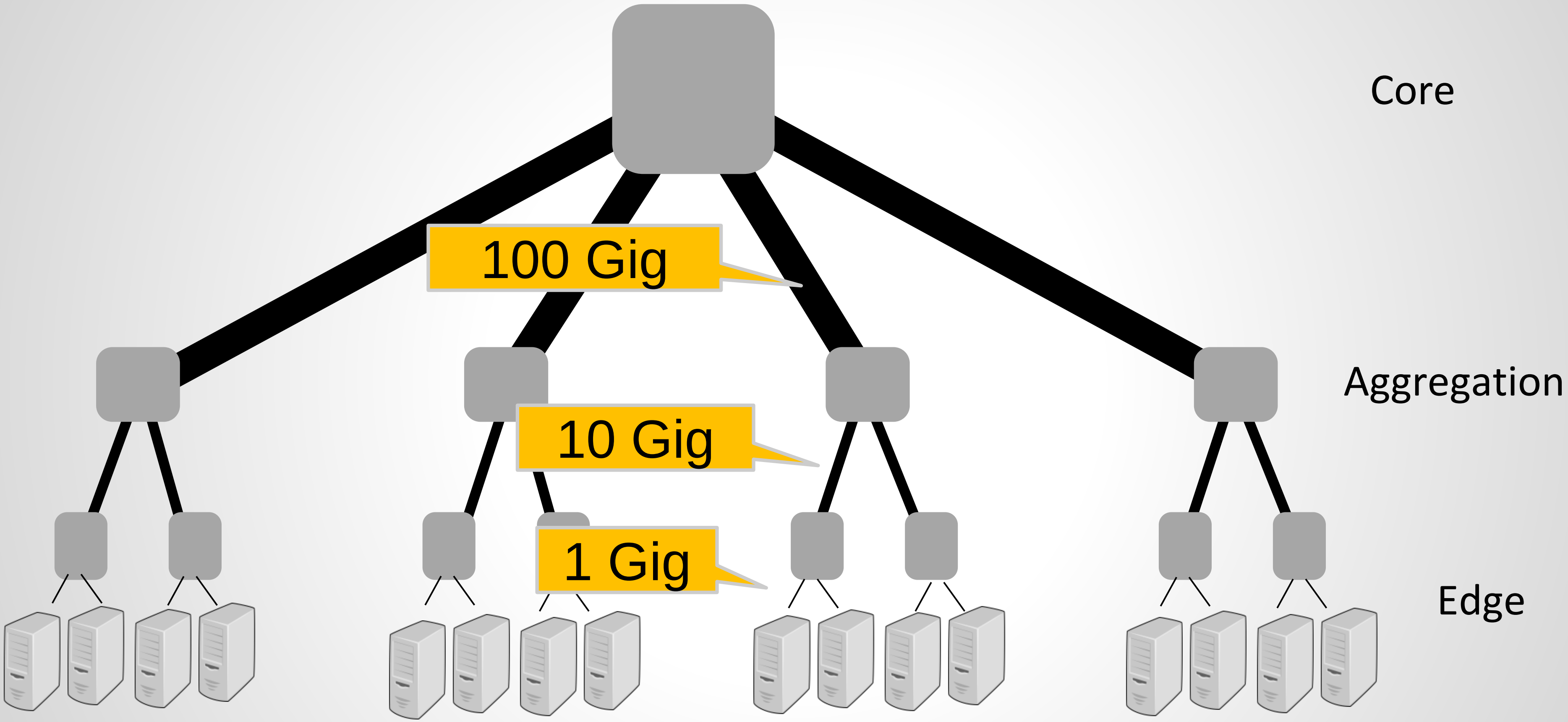
The Fat-Tree Topology: Theory



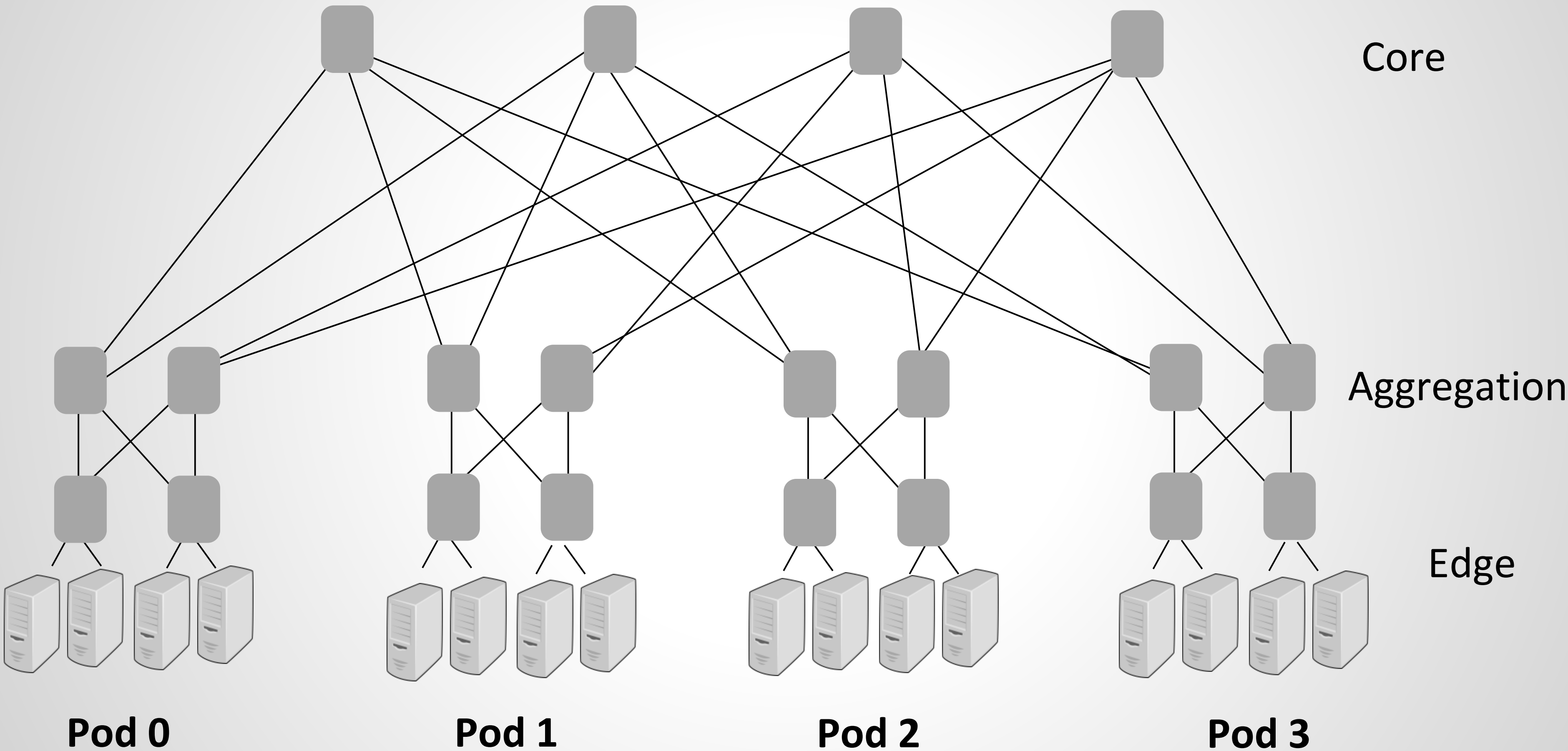
The Fat-Tree Topology: Theory



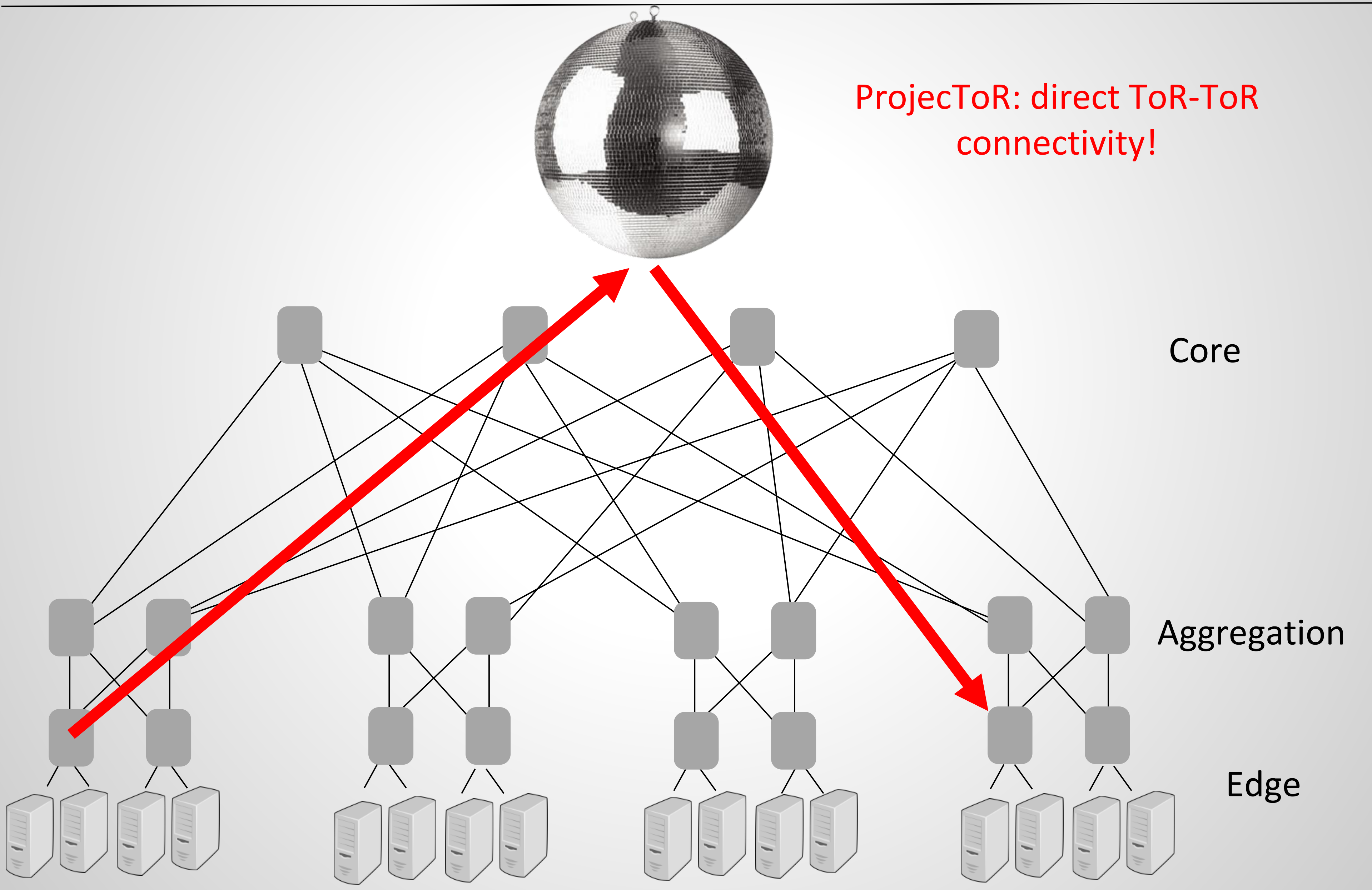
The Fat-Tree Topology: Theory



The Fat-Tree Topology: Practice



The Fat-Tree Topology: Future?



Reconfigurable Networks

- Reconfigurable interconnects, e.g., based on optical circuit switches, 60 GHz wireless, and free-space optics, allow to directly connect frequently communicating pairs of racks (e.g., using digital micromirror devices)
- Emerging technologies: ProjecToR, REACToR, Flyways, Mirror, Firefly, etc. allow to reconfigure the (physical) topology of communication networks at runtime
- Attractive: real communication patterns are far from "all-to-all", but usually feature much structure and are sparse

Robustness aspects not studied yet!

First Insights: Model 1 «Bounded Network Design»

Given a demand matrix…

$$
\begin{array}{c|c}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\begin{bmatrix}
0 & \frac{2}{65} & \frac{1}{13} & \frac{1}{65} & \frac{1}{65} & \frac{2}{65} & \frac{3}{65} \\
\frac{2}{65} & 0 & \frac{1}{65} & 0 & 0 & 0 & \frac{2}{65} \\
\frac{1}{13} & \frac{1}{65} & 0 & \frac{2}{65} & 0 & 0 & \frac{1}{13} \\
\frac{1}{65} & 0 & \frac{2}{65} & 0 & \frac{4}{65} & 0 & 0 \\
\frac{1}{65} & 0 & \frac{3}{65} & \frac{4}{65} & 0 & 0 & 0 \\
\frac{2}{65} & 0 & 0 & 0 & 0 & 0 & \frac{3}{65} \\
\frac{3}{65} & \frac{2}{65} & \frac{1}{13} & 0 & 0 & \frac{3}{65} & 0
\end{bmatrix}
\end{array}
$$

… find a network of max degree

which minimizes the expected path length!

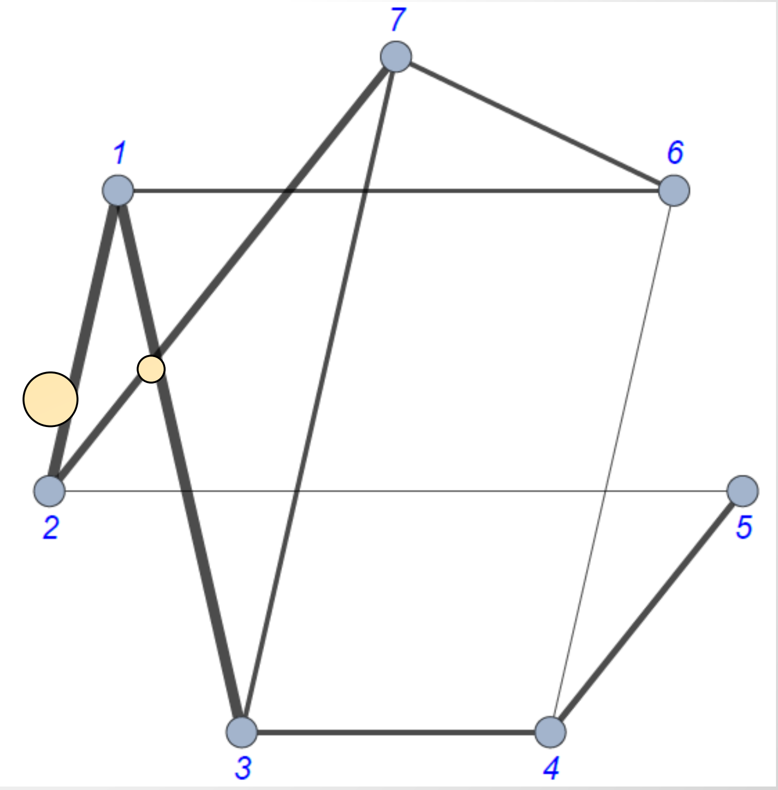
Ideas?
Subgraph?

First Insights: Model 1 «Bounded Network Design»

Given a demand matrix…

	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

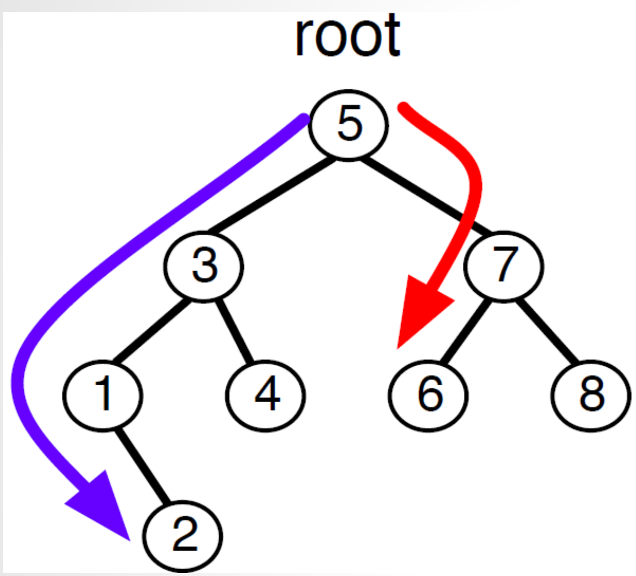
… find a network of max degree which minimizes the expected path length!

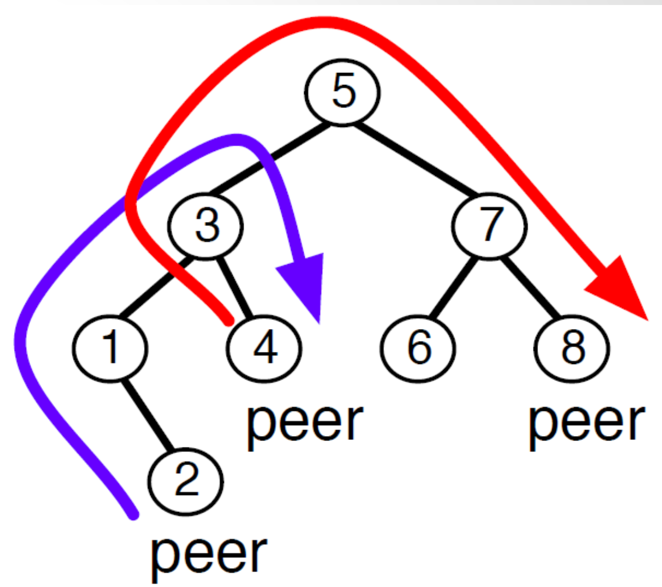


Can first build subgraph and then “rewire edges”, make it connected and low degree!

First Insights: Model 2 «SplayNets»

Distributed generalization of self-adjusting data structures, e.g., splay tree binary search trees:





Splay Tree

Move-to-front

(Move-to-root)

SplayNet

Move-to-LCA

Open Questions

- Bounded degree network design for arbitrary demand matrices?
- Robust bounded degree network design?
- Static optimality, dynamic optimality, static finger, dynamic finger for SplayNets?
- Robust SplayNets?
- From Clos to WAN?

Further Reading

SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**ToN**), Volume 24, Issue 3, 2016.

Demand-Aware Network Designs of Bounded Degree
Chen Avin, Kaushik Mondal, and Stefan Schmid.
31st International Symposium on Distributed Computing (**DISC**), Vienna, Austria, October 2017.

Conclusion

- SDN introduces many flexibilities
- But also new challenges
  - How to exploit flexibilities algorithmically?
  - How to deal with remote controller(s)?
- Another grand challenge: reconfigurable topologies for datacenter and WAN (amortized and competitive analysis!)

Algorithms for flow rerouting:

Can't Touch This: Consistent Network Updates for Multiple Policies
Szymon Dudycz, Arne Ludwig, and Stefan Schmid.
46th IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Toulouse, France, June 2016.

loop-freedom
multiple policies

Transiently Secure Network Updates
Arne Ludwig, Szymon Dudycz, Matthias Rost, and Stefan Schmid.
42nd ACM **SIGMETRICS**, Antibes Juan-les-Pins, France, June 2016.

waypointing

Scheduling Loop-free Network Updates: It's Good to Relax!
Arne Ludwig, Jan Marcinkowski, and Stefan Schmid.
ACM Symposium on Principles of Distributed Computing (**PODC**), Donostia-San Sebastian, Spain, July 2015.

loop-freedom

Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies
Arne Ludwig, Matthias Rost, Damien Foucard, and Stefan Schmid.
13th ACM Workshop on Hot Topics in Networks (**HotNets**), Los Angeles, California, USA, October 2014.

waypointing

Congestion-Free Rerouting of Flows on DAGs
Saeed Akhoondian Amiri, Szymon Dudycz, Stefan Schmid, and Sebastian Wiederrecht.
ArXiv Technical Report, November 2016.

capacity constraints

Survey of Consistent Network Updates
Klaus-Tycho Foerster, Stefan Schmid, and Stefano Vissicchio.
ArXiv Technical Report, September 2016.

survey

Security of the data plane:

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.
See also CVE-2015-7516.

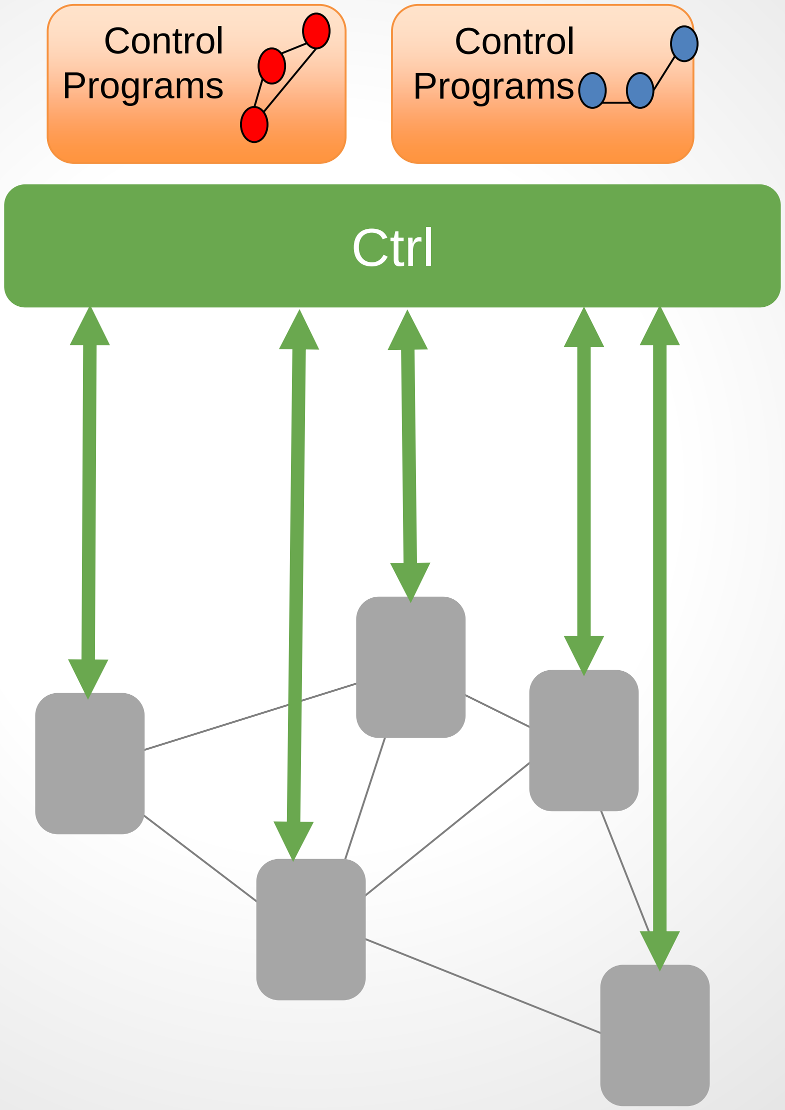
teleportation

attacking the cloud

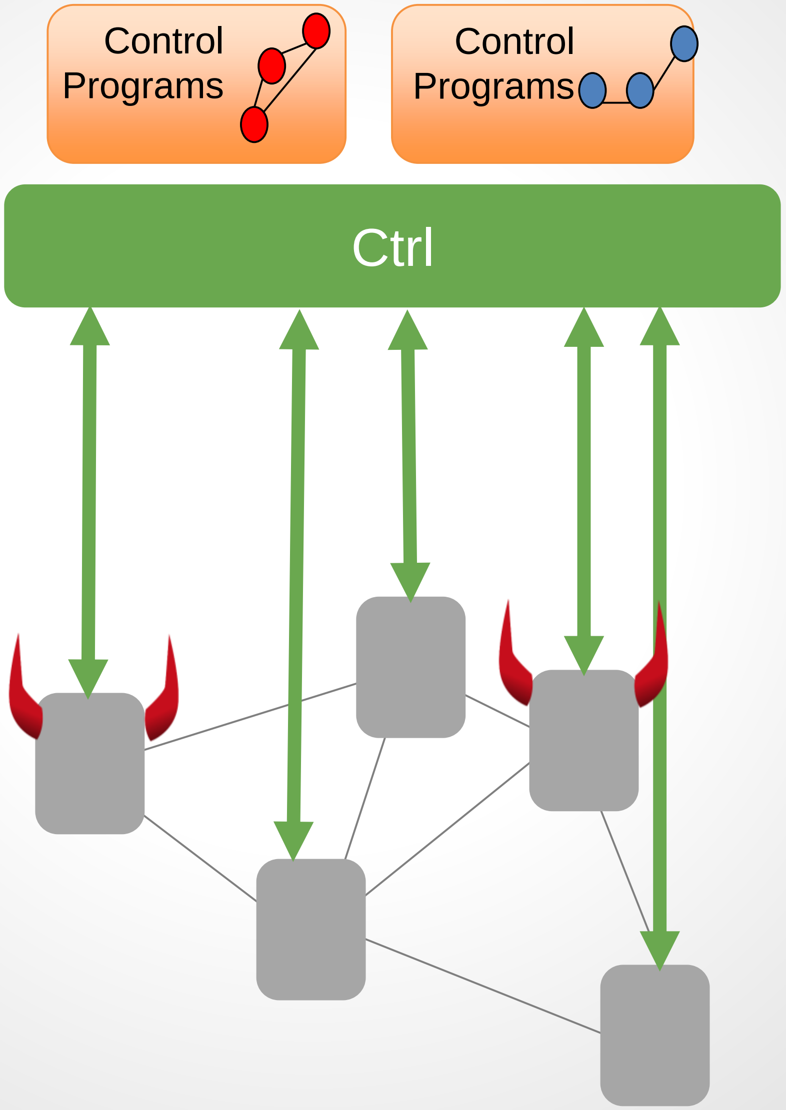
Reigns to the Cloud: Compromising Cloud Systems via the Data Plane
Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid.
ArXiv Technical Report, October 2016.

Backup Slides

Let's talk about security!

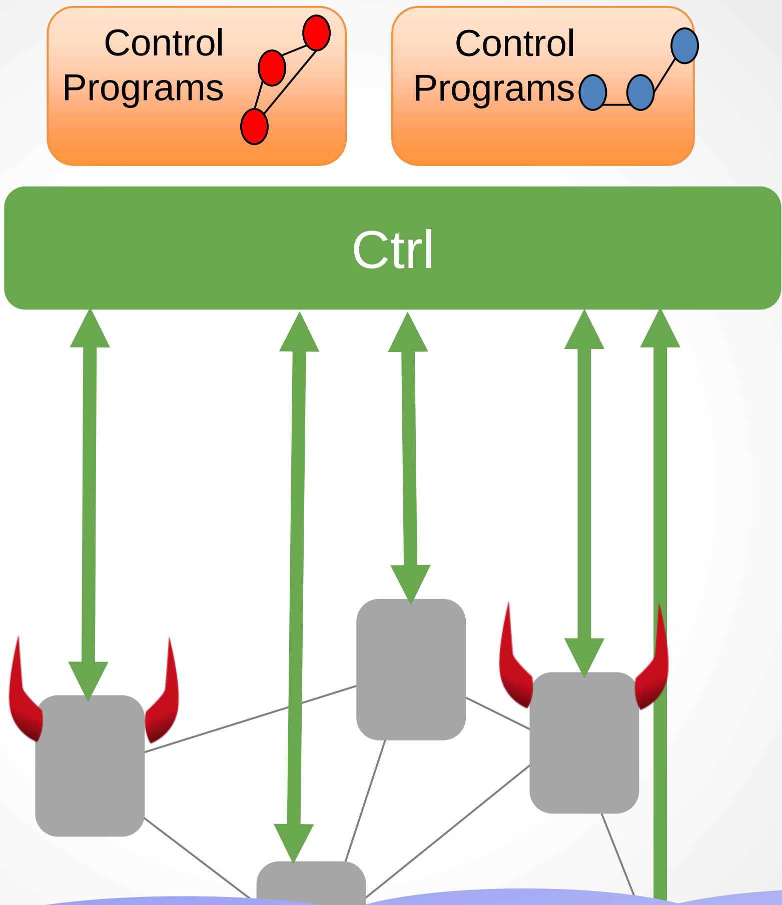


Let’s talk about security!



In particular:
malicious switches
(less studied).

Let’s talk about security!



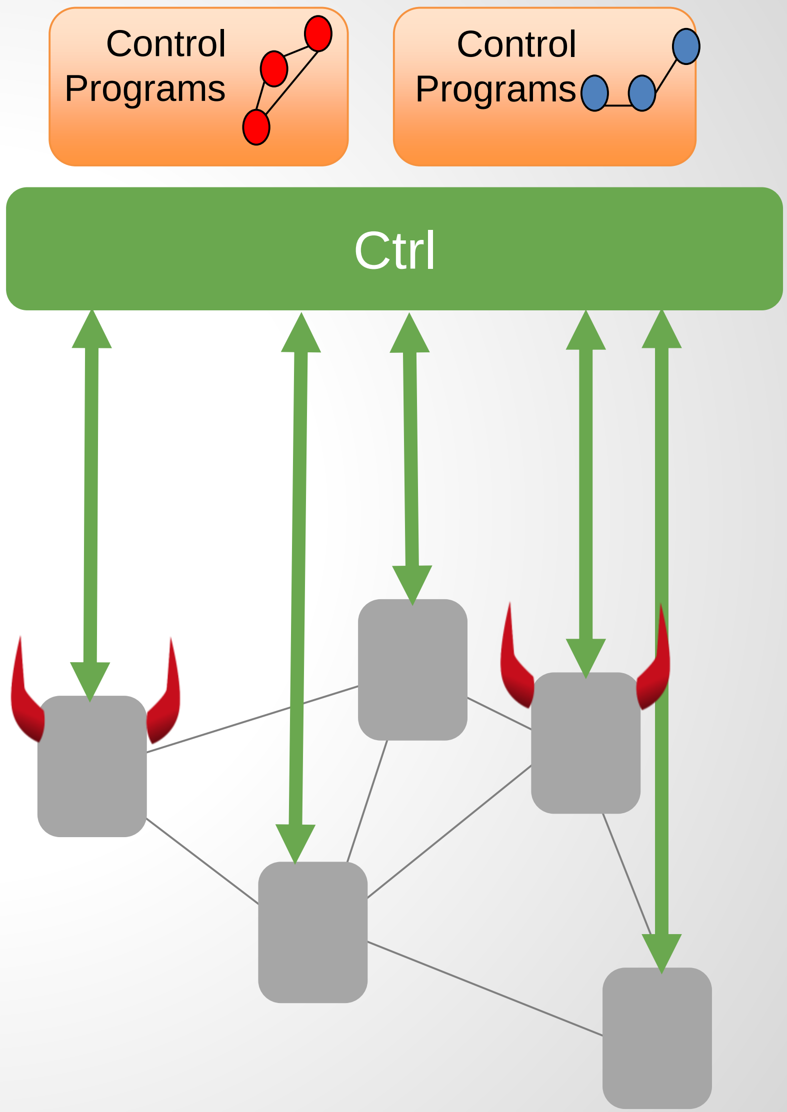
In particular: malicious switches (less studied).

Note: Governments etc. don’t have resources to build their own trusted hardware.

Let's talk about security!

The case for insecure data planes: many incidents

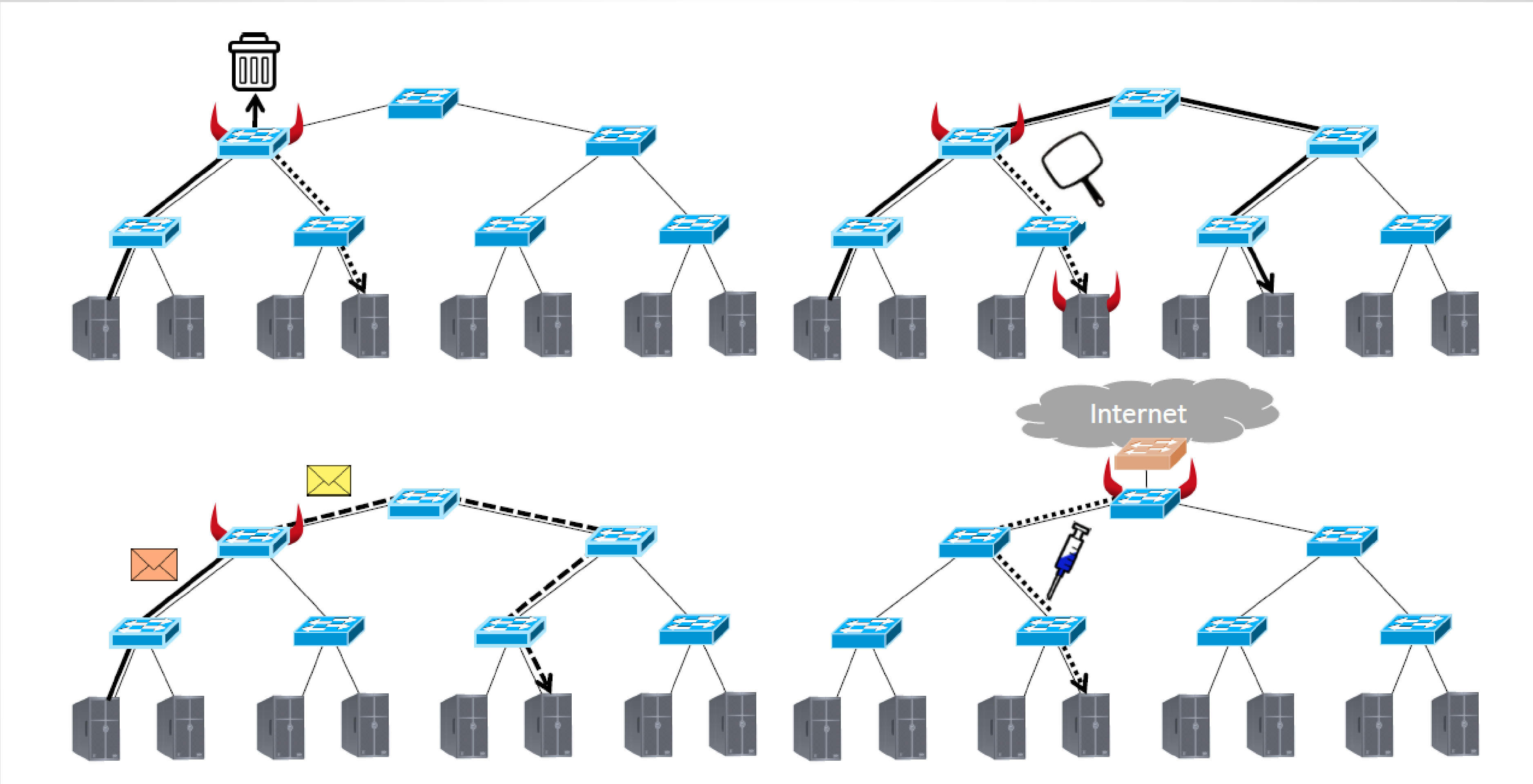
- ❑ Attackers have compromised routers
- ❑ Compromised routers are traded underground
- ❑ Vendors have left backdoors open
- ❑ National security agencies can bug network equipment
- ❑ …



What a malicious switch could do:

1 drop/reroute/exfiltrate

2 mirror



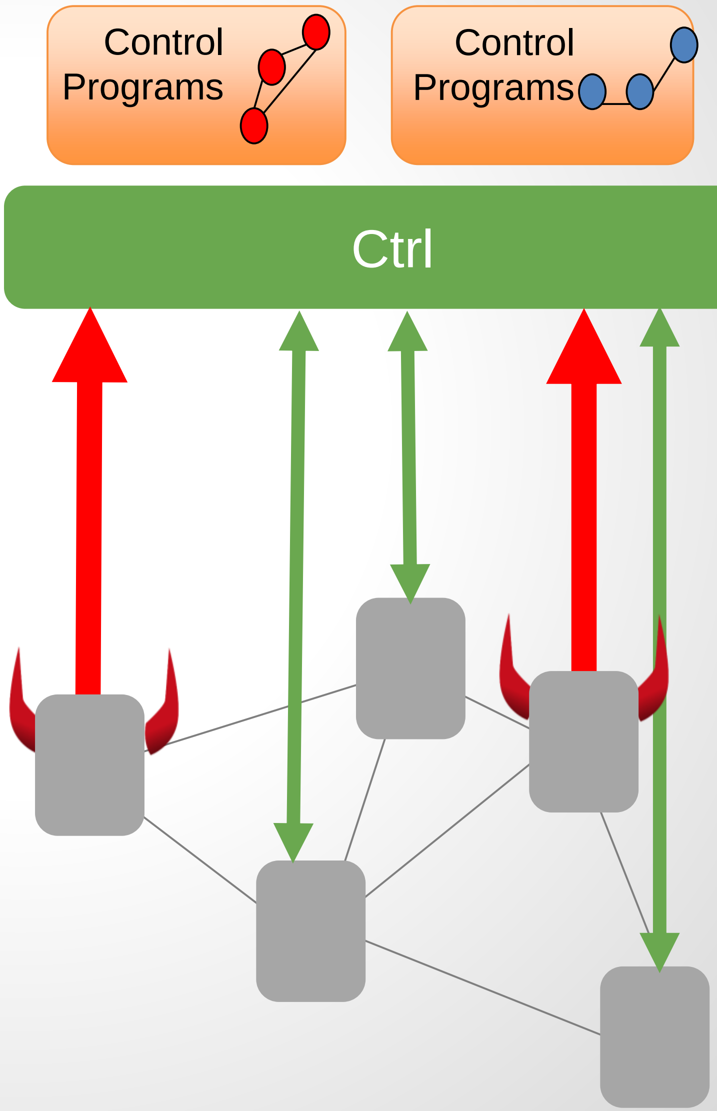
3 modify

4 inject

More and New Attacks in SDN

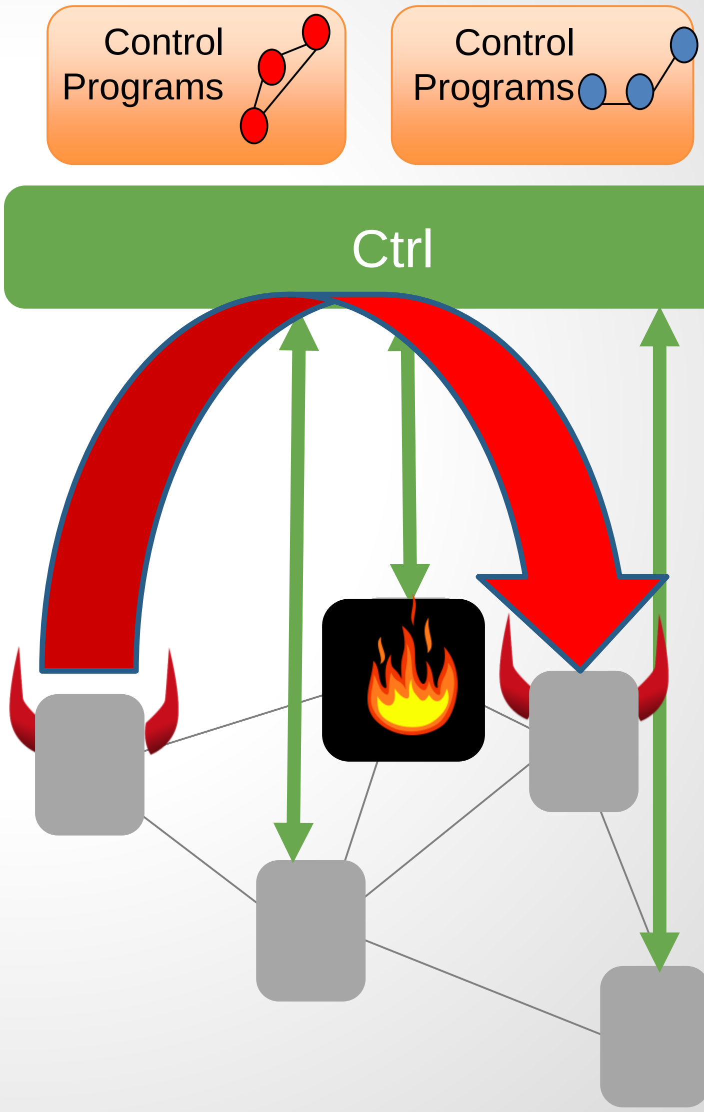
New attack vector:

- DoS on controller
- Harms availability
- E.g., force other switches into default behavior



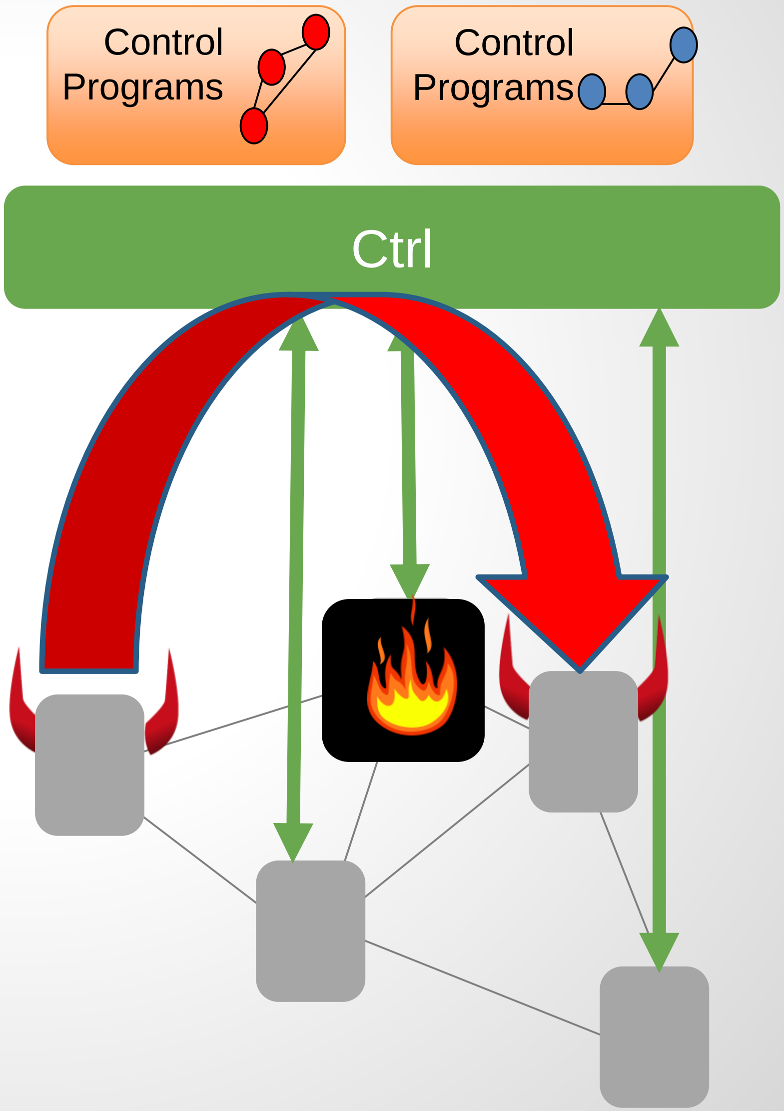
Another New SDN Attack: Teleportation

- Idea: exploit controller to communicate information: «Teleportation»



Another New SDN Attack: Teleportation

- Idea: exploit controller to communicate information: «Teleportation»
- Controller reacts to switch events (packet-ins) by sending flowmods/packet-outs/… etc.: can be exploited to transmit information
- E.g., in MAC learning: src MAC 0xBADDAD
- Can also modulate information implicitly (e.g., frequency of packetins)
- E.g.: covert communication, bypass firewall, coordinate attack

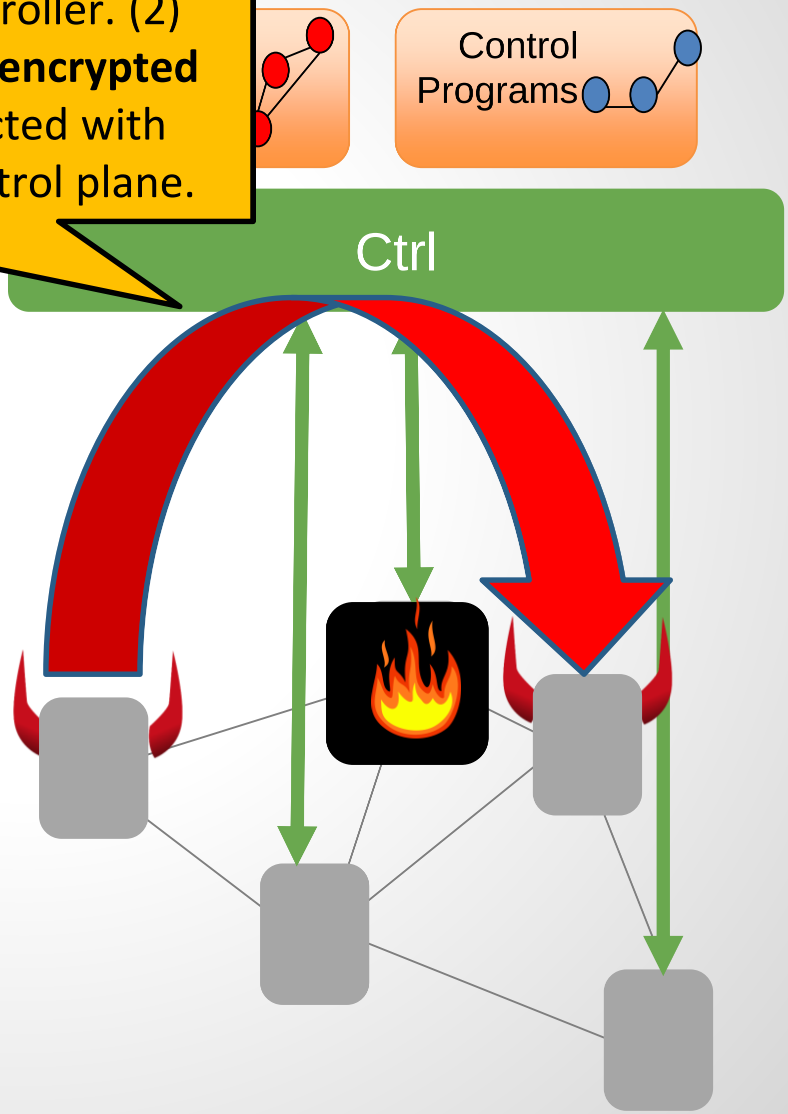


Another New SDN Attack: Teleportation

Difficult to detect: (1) The teleported information follows the **normal traffic pattern** of control communication, indirectly between any switch and the controller. (2) Teleportation channel is inside the typically **encrypted OpenFlow** channel. Cannot easily be detected with modern IDS, even if they operate in the control plane.

events (packet-ins) by sending flowmods/packet-outs/… etc.: can be exploited to transmit information

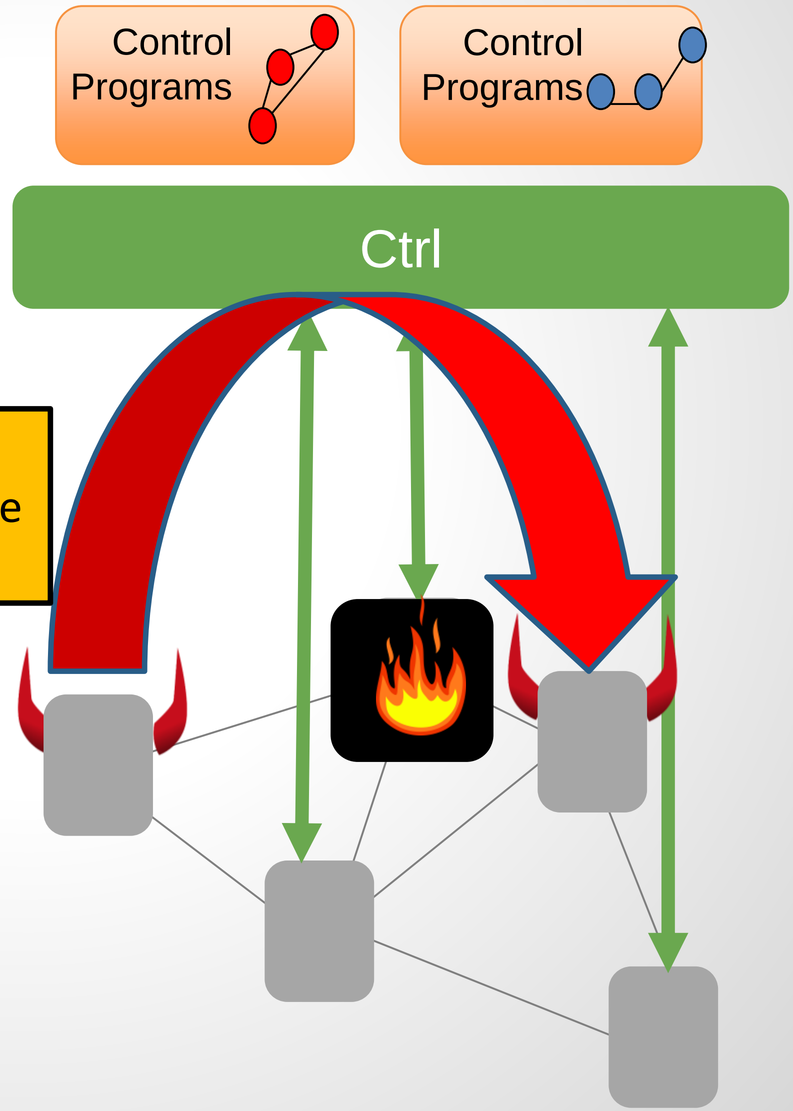
- E.g., in MAC learning: src MAC 0xBADDAD
- Can also modulate information implicitly (e.g., frequency of packetins)
- E.g.: covert communication, bypass firewall, coordinate attack



Another New SDN Attack: Teleportation

- Idea: exploit controller to communicate information: «Teleportation»
- Controller reacts to switch events (packet-ins) by sending flowmods/packet-outs/… etc.: can be exploited to transmit informati
- E.g., in MA 0xBADDAD
- Can also modulate information implicitly (e.g., frequency of packetins)
- E.g.: covert communication, bypass firewall, coordinate attack

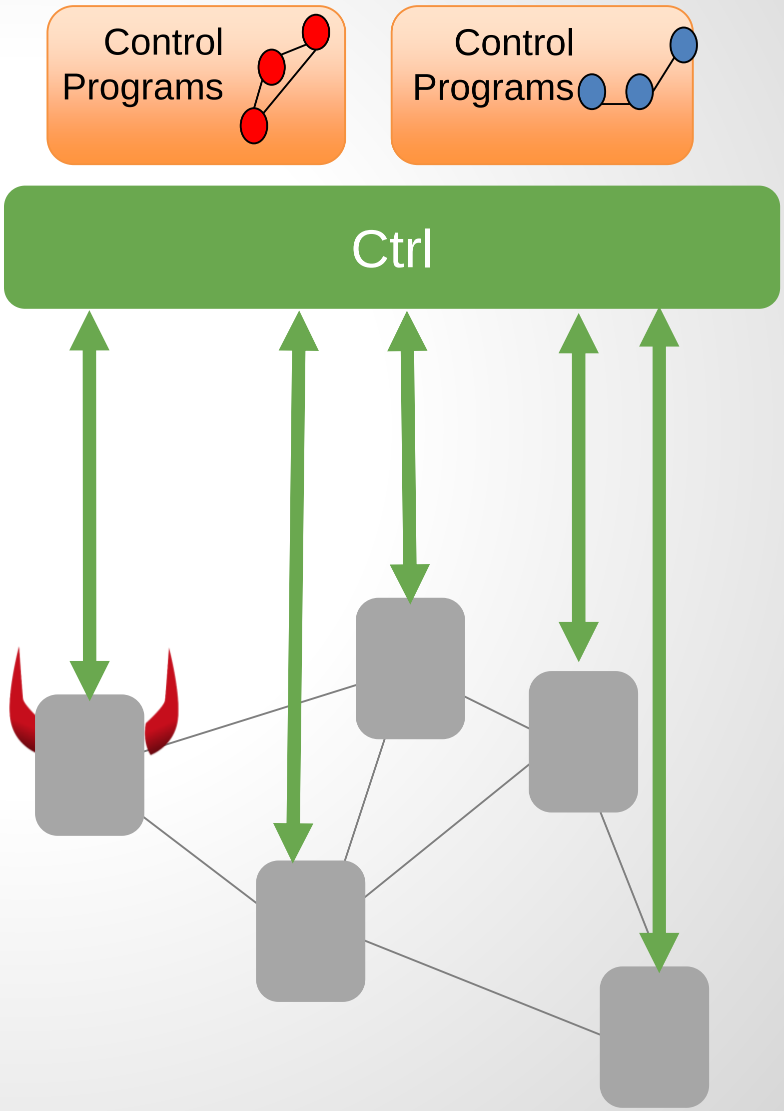
E.g., 2 switches try to use the same DPID, exploit pave path technique, etc.



Another Front: Virtualized Switches

Attack vector:

- The virtualized data plane



Further Reading

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.

Another Front: Virtualized Switches

Attack vector:

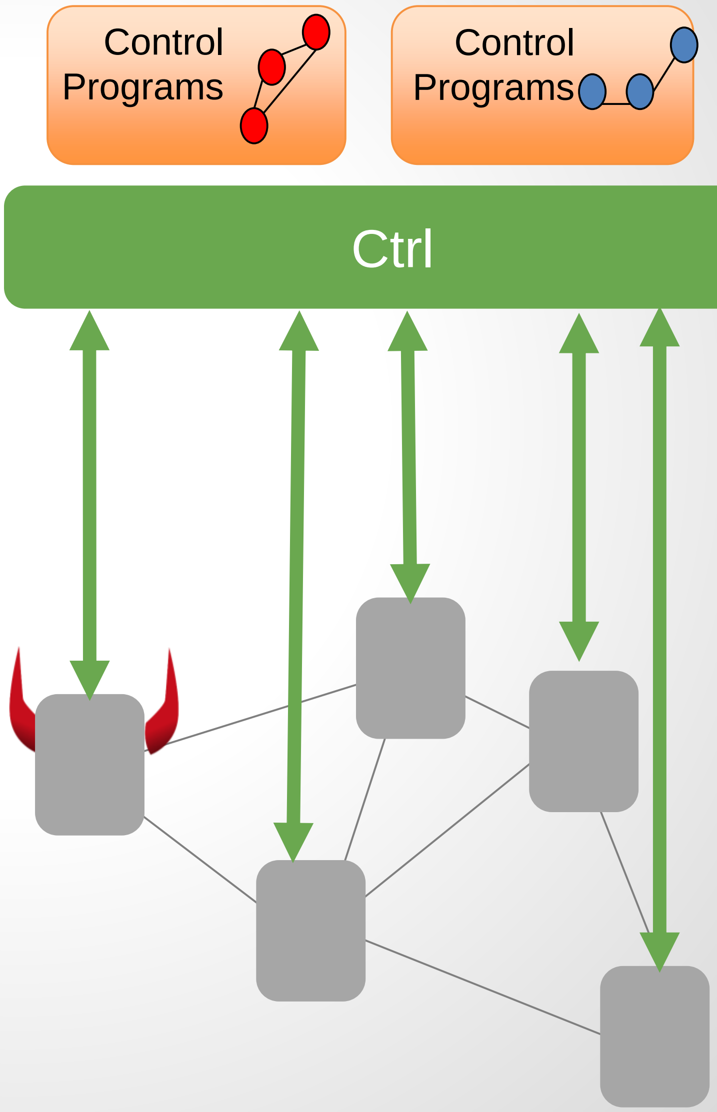
- ❑ The virtualized data plane

Background:

- ❑ Packet processing and other network functions are more and more virtualized
- ❑ E.g., runing on servers at the edge of the datacenter
- ❑ Example: OVS

Advantage:

- ❑ Cheap and performance ok!
- ❑ Fast and easy deployment



Security Challenges: Insecure Dataplane

Attack vector:

- ❑ The virtualized data plane

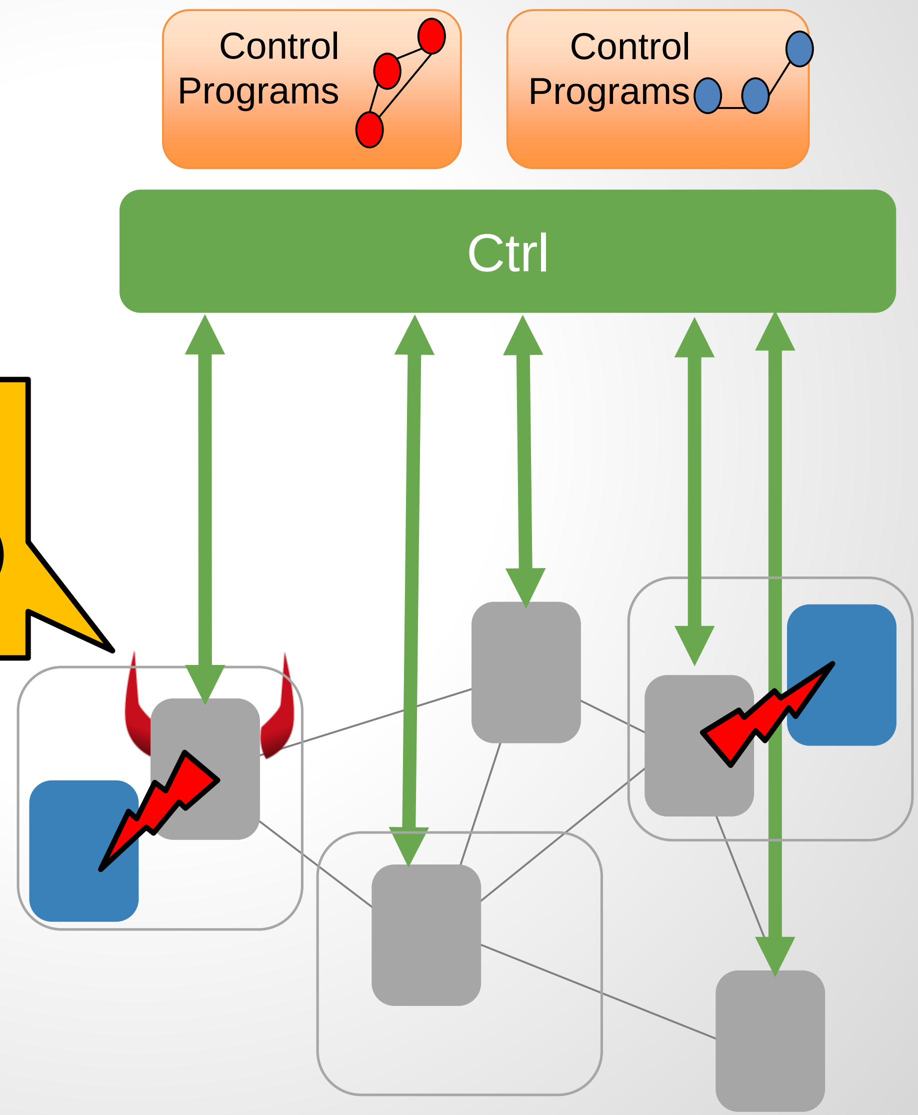
Background:

- ❑ Packet processing and other network functions are more and more virtua
- ❑ E.g., the e
- ❑ Example: OVS

New vulnerability: collocation. Switches run with evelated (root) priviledges.

Advantage:

- ❑ Cheap and performance ok!
- ❑ Fast and easy deployment



Security Challenges: Insecure Dataplane

Attack vector:

- ❑ The virtualized data plane

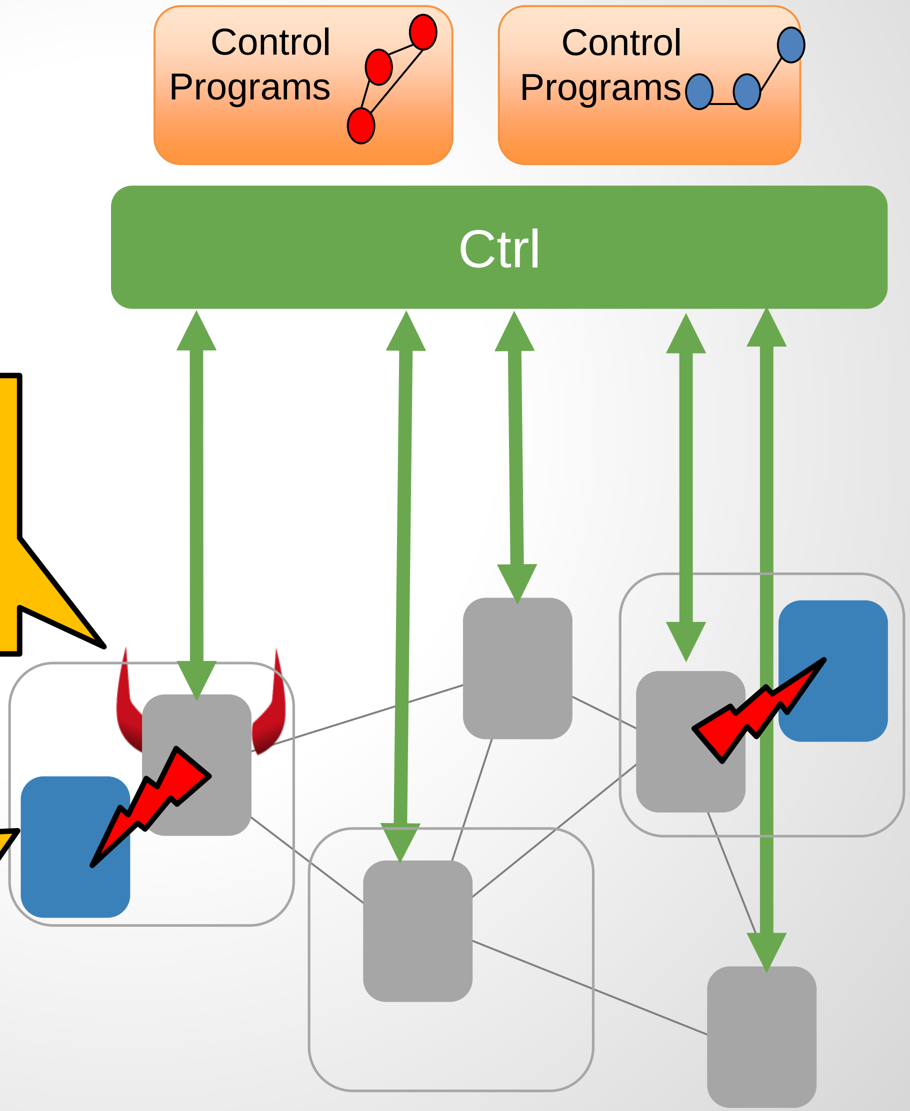
Background:

- ❑ Packet processing and other network functions are more and more virtua
- ❑ E.g., the e
- ❑ Example: OVS

New vulnerability: collocation. Switches run with evelated (root) priviledges.

Advantage:

Collocated with e.g., controllers, hypervisors, guest VMs, VM image and network management, identity management (of admins and tenants), etc.



A Case Study: OVS

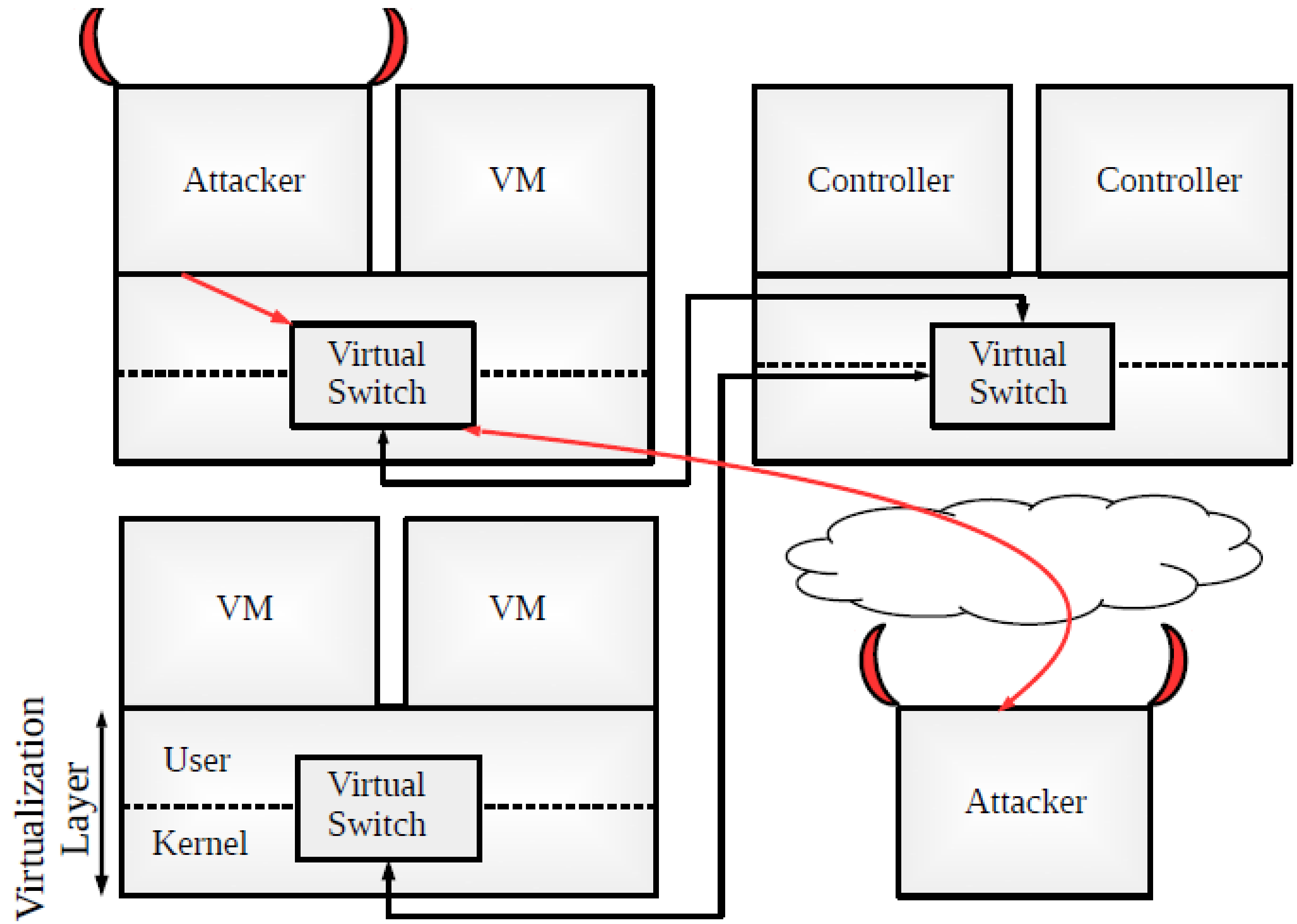
- OVS: a production quality switch, widely deployed in the Cloud
- After fuzzing just 2% of the code, found major vulnerabilities:
  - E.g., two stack overflows when malformed MPLS packets are parsed
- These vulnerabilities can easily be weaponized:
  - Can be exploited for arbitrary remote code execution
  - E.g., our «reign worm» compromised cloud setups within 100s
- Significance
  - It is often believed that only state-level attackers (with, e.g., control over the vendor's supply chain) can compromise the data plane
  - Virtualized data planes can be exploited by very simple, low-budget attackers: e.g., by renting a VM in the cloud and sending a single malformed MPLS packet

The Reign Worm

Exploits 4 problems:

1. **Security assumptions:** Virtual switches often run with elevated (root) priviledges by design.
2. **Collocation:** virtual switchs reside in virtualized servers (Dom0), and are hence collocated with other and possibly critical cloud software, including controller software
3. **Logical centralization:** the control of data plane elements is often outsourced to a centralized software. The corresponding bidirectional communication channels can be exploited to spread the worm further.
4. **Support for extended protocol parsers:** Virtual switches provide functionality which goes beyond basic protocol locations of normal switches (e.g., handling MPLS in non-standard manner)

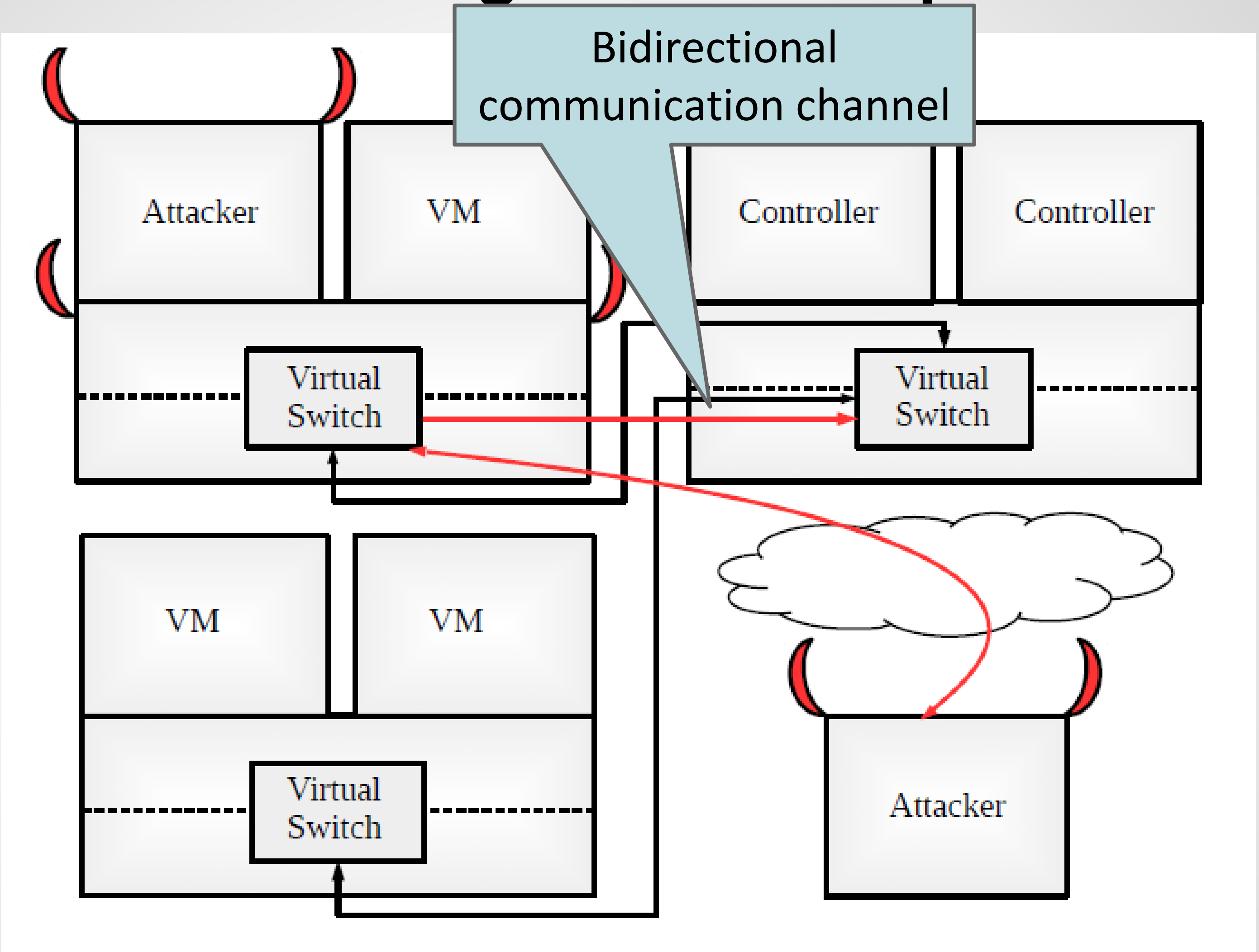
The Reign Worm: Step 1



(a) Step 1.

Attacker VM sends a malicious packet that compromises its server, giving the remote attacker control of the server.

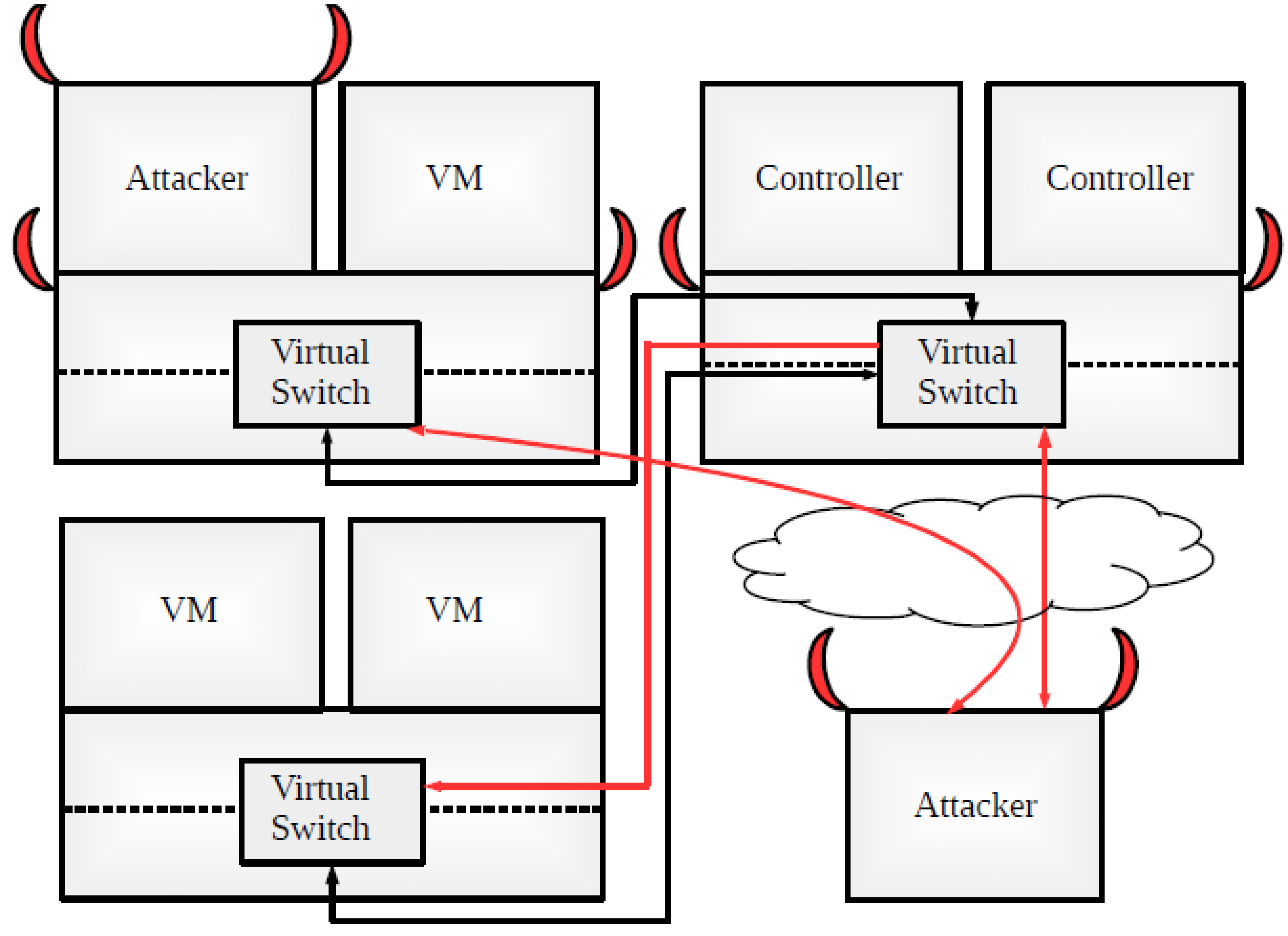
The Reign Worm: Step 2



(b) Step 2.

Attacker controlled server compromises the controllers' server, giving the remote attacker control of the controllers' server.

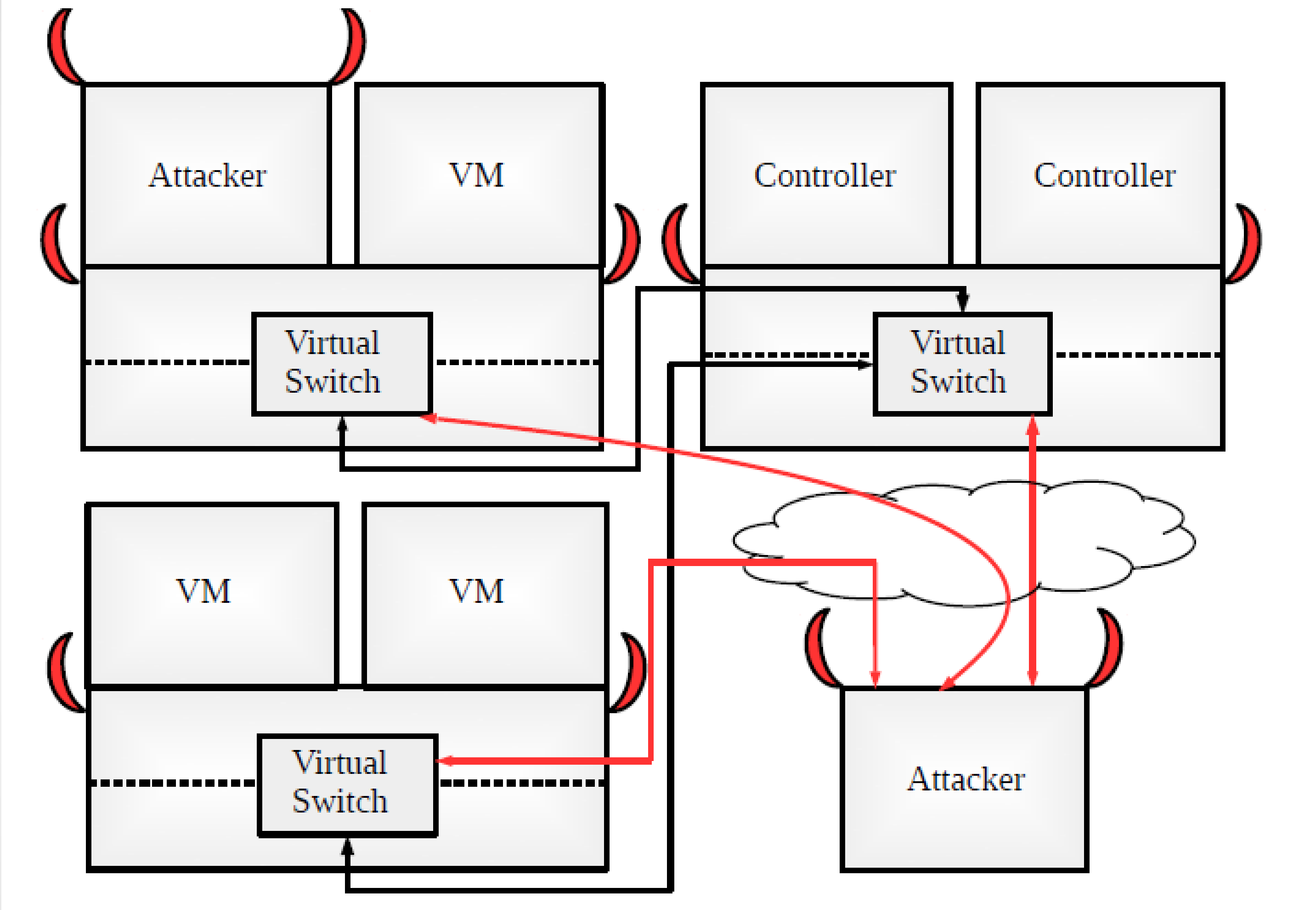
The Reign Worm: Step 3



(c) Step 3.

The compromised controllers' server propagates the worm to the remaining uncompromised server.

The Reign Worm: Step 4



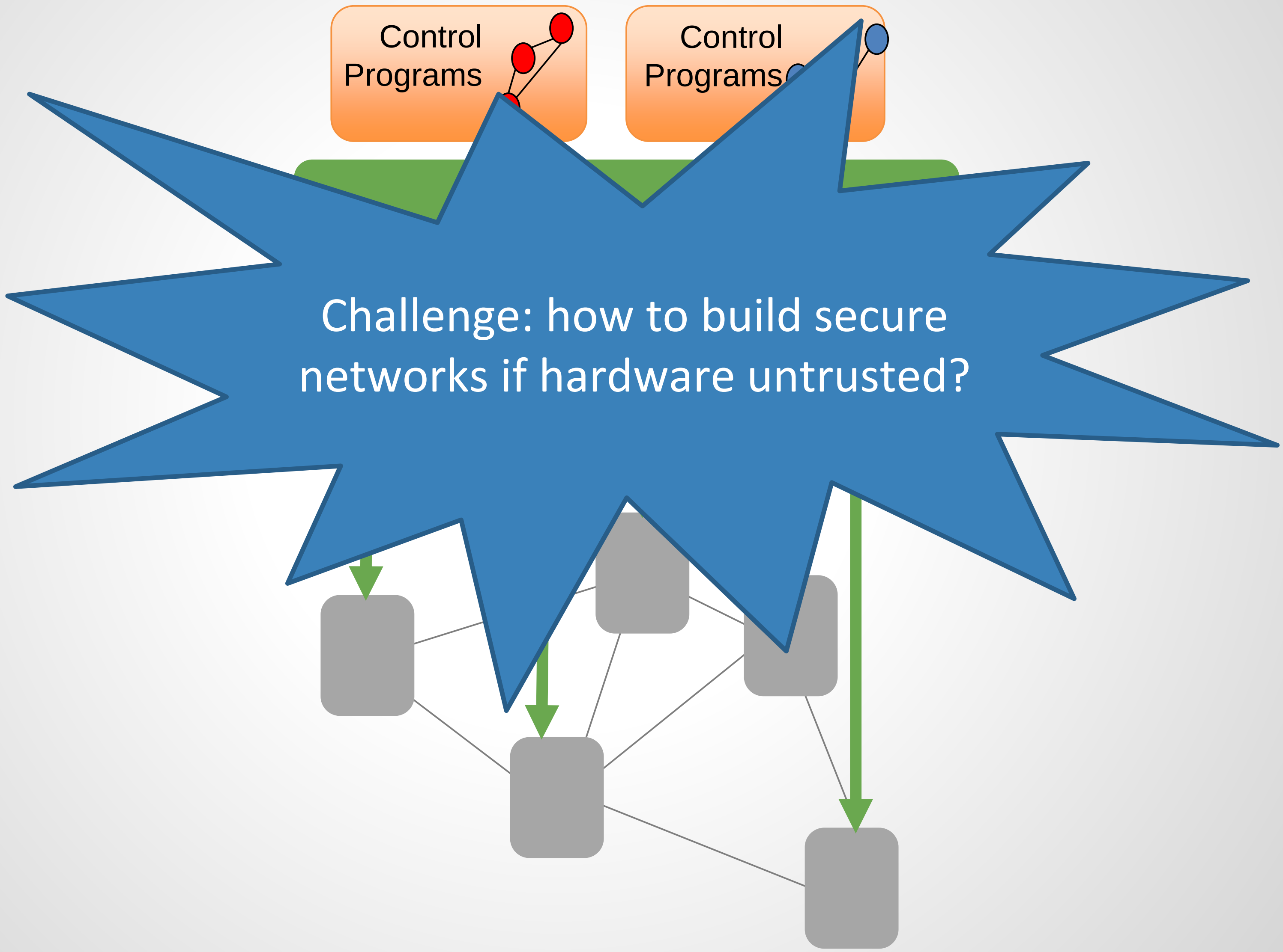
(d) Step 4.

All the servers are controlled by the remote attacker.

Further Reading

[Reigns to the Cloud: Compromising Cloud Systems via the Data Plane](#)
Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid. ArXiv Technical Report, October 2016.

Let’s talk about security!

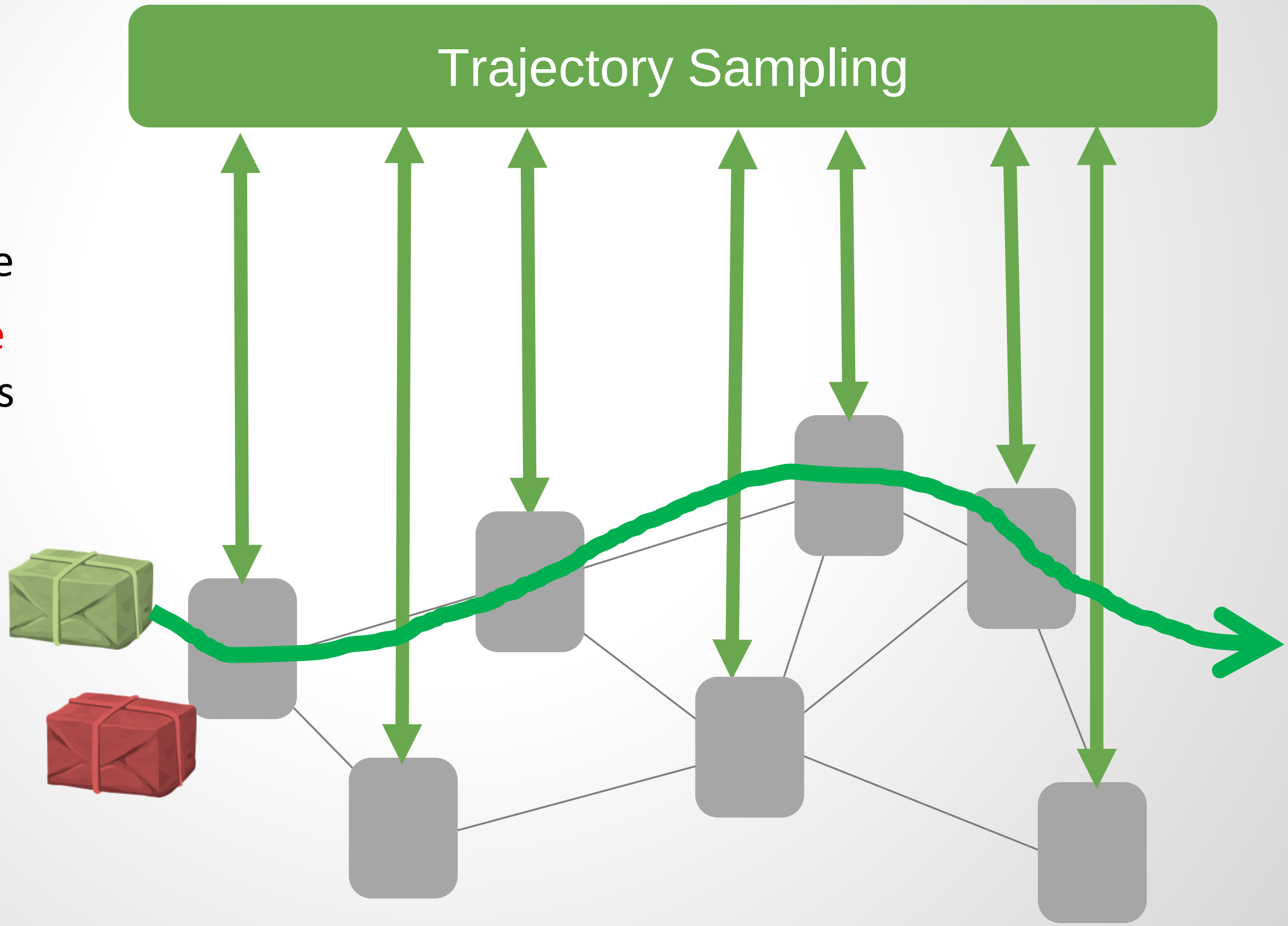


Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling

Principle:

- Sample subset of packets (based on hash value) anytime
- Get complete route for sampled packets
- Efficient: sampling

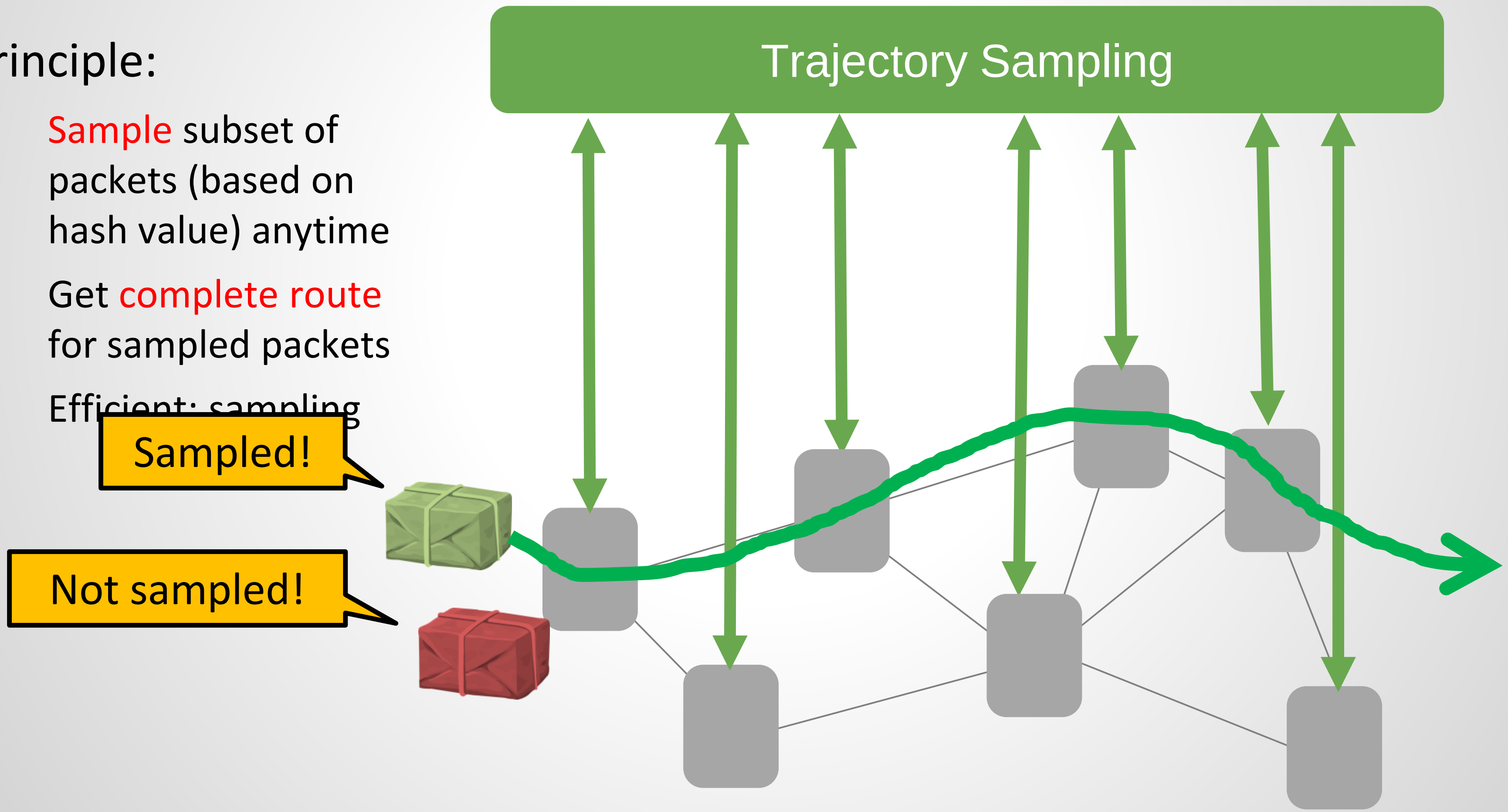


Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling

Principle:

- Sample subset of packets (based on hash value) anytime
- Get complete route for sampled packets
- Efficient: sampling

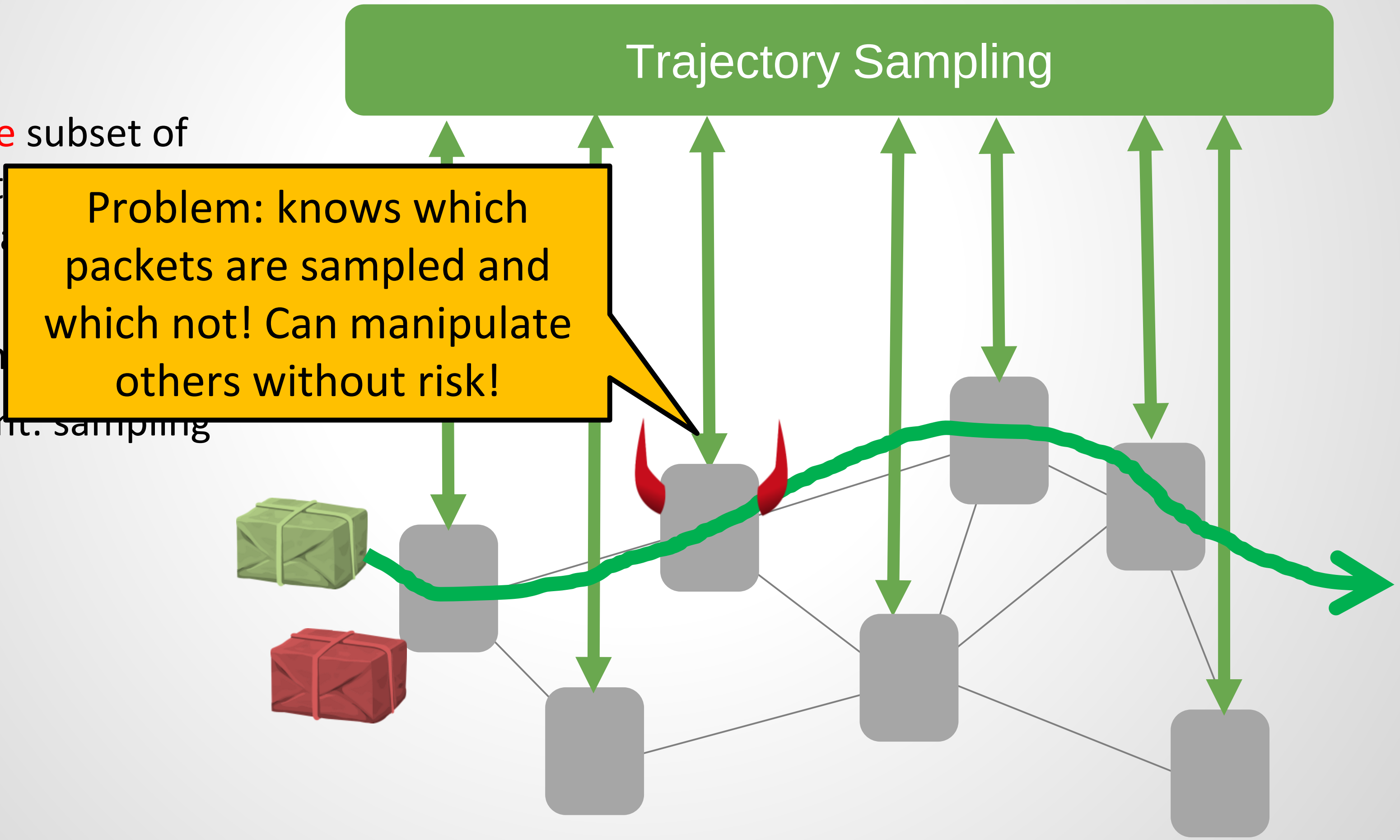


Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling

Principle:

- Sample subset of packet… hash v…
- Get co… for san…
- Efficient: sampling

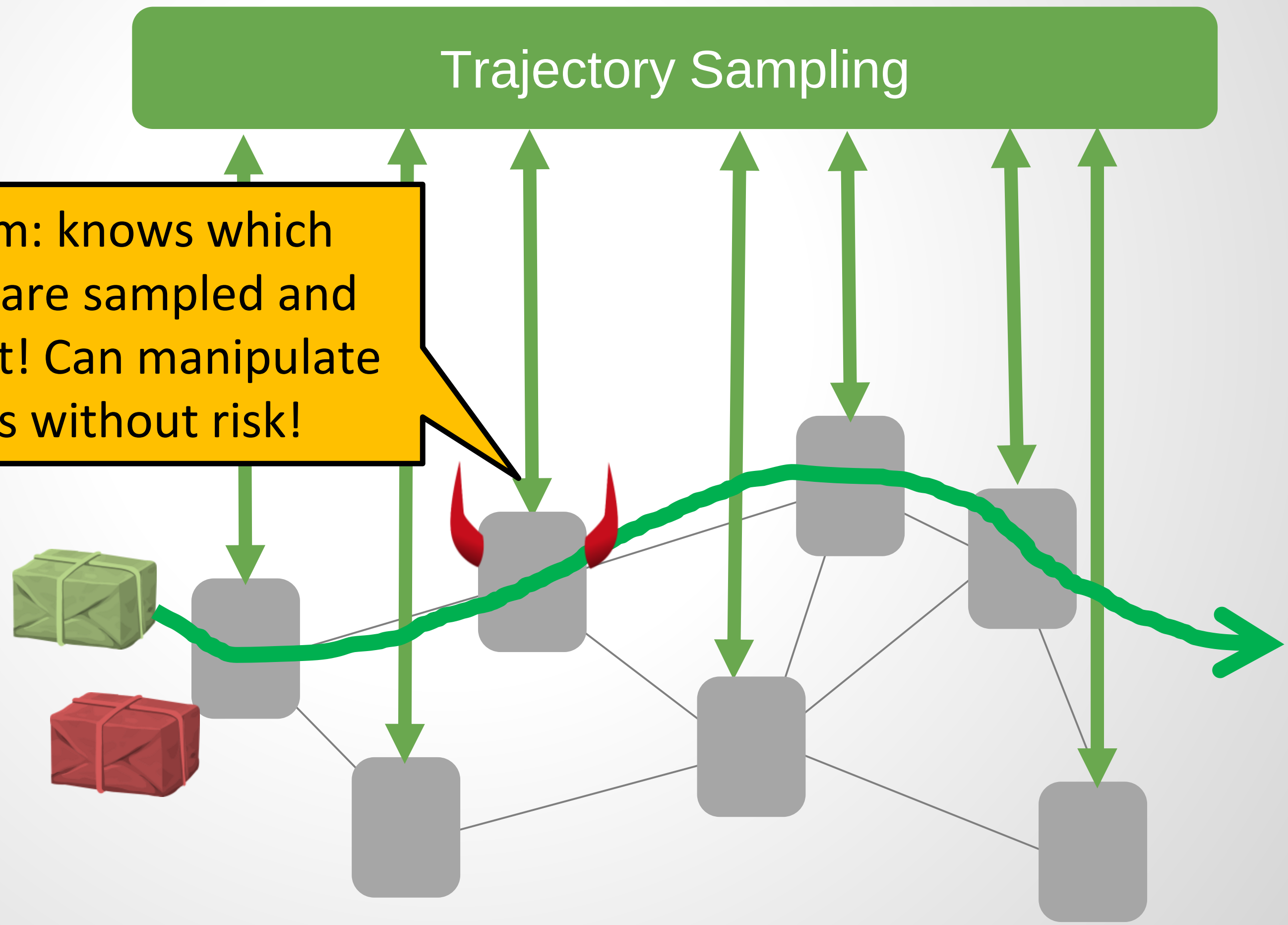


Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling

Principle:

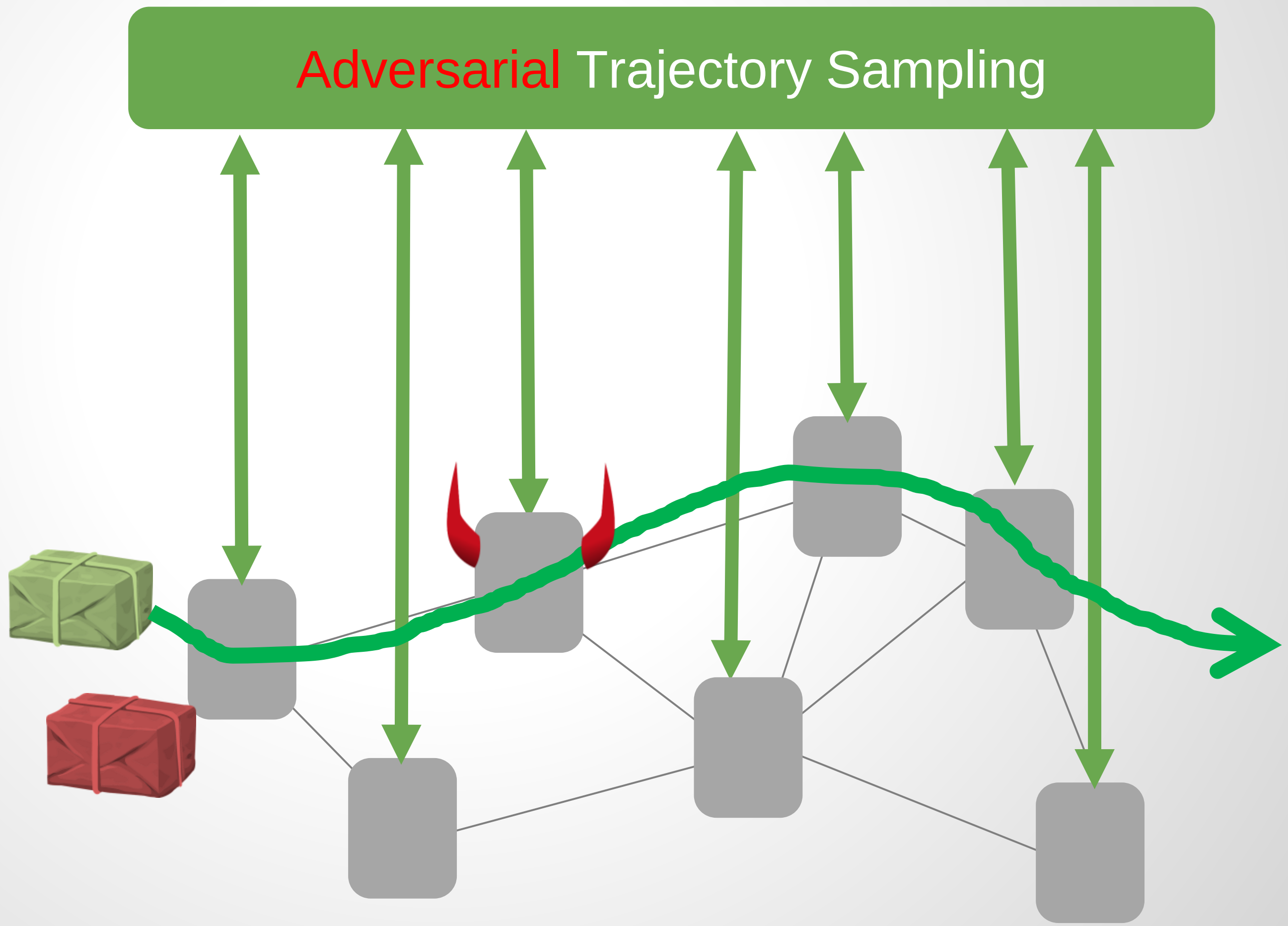
- Sample subset of packet… hash v…
- Get co… for san…
- Efficient: sampling



How to make trajectory sampling secure to malicious switches?

Opportunity: Adversarial Trajectory Sampling

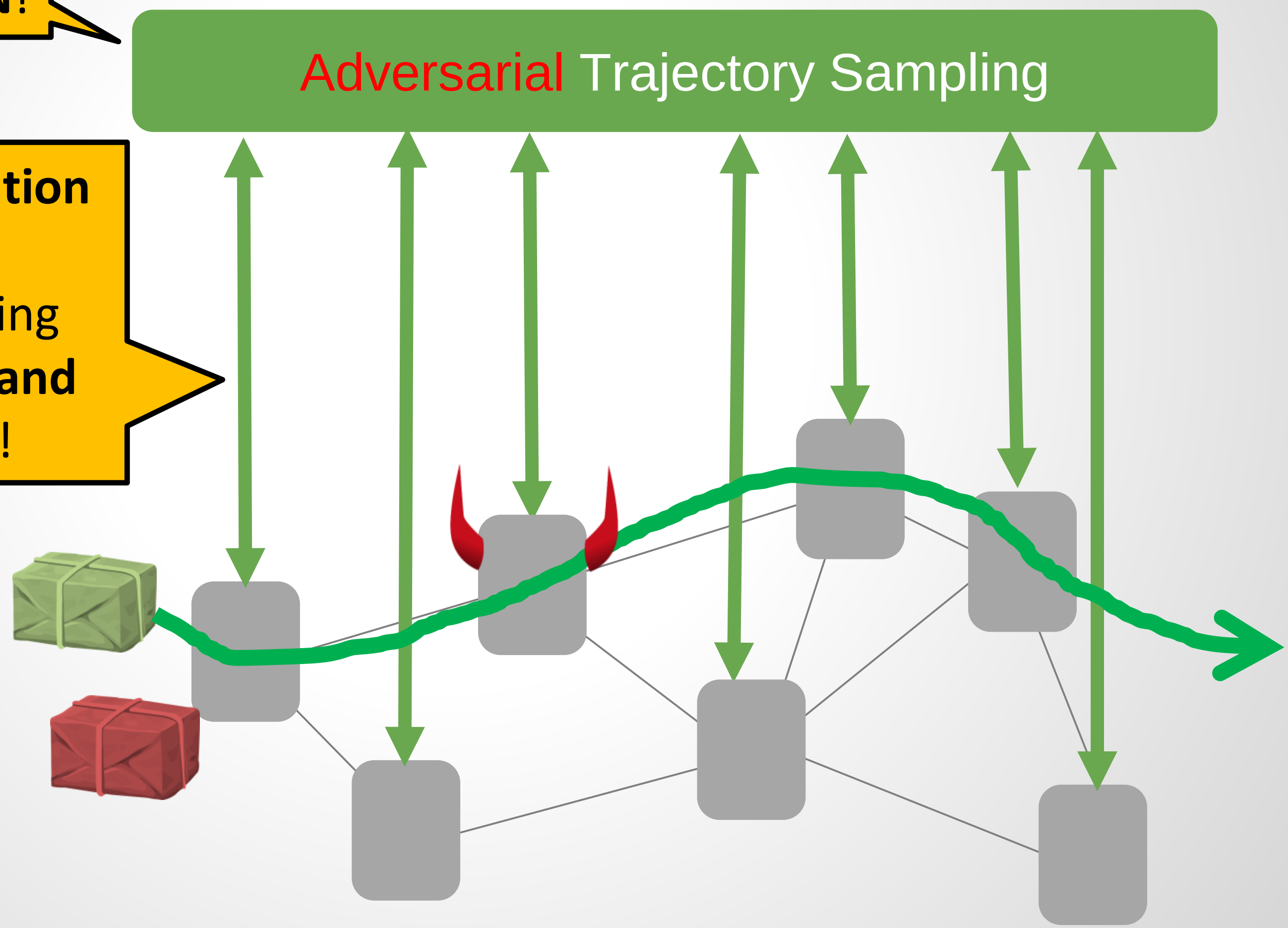
- Classic tool to monitor packet routes: trajectory sampling



Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling

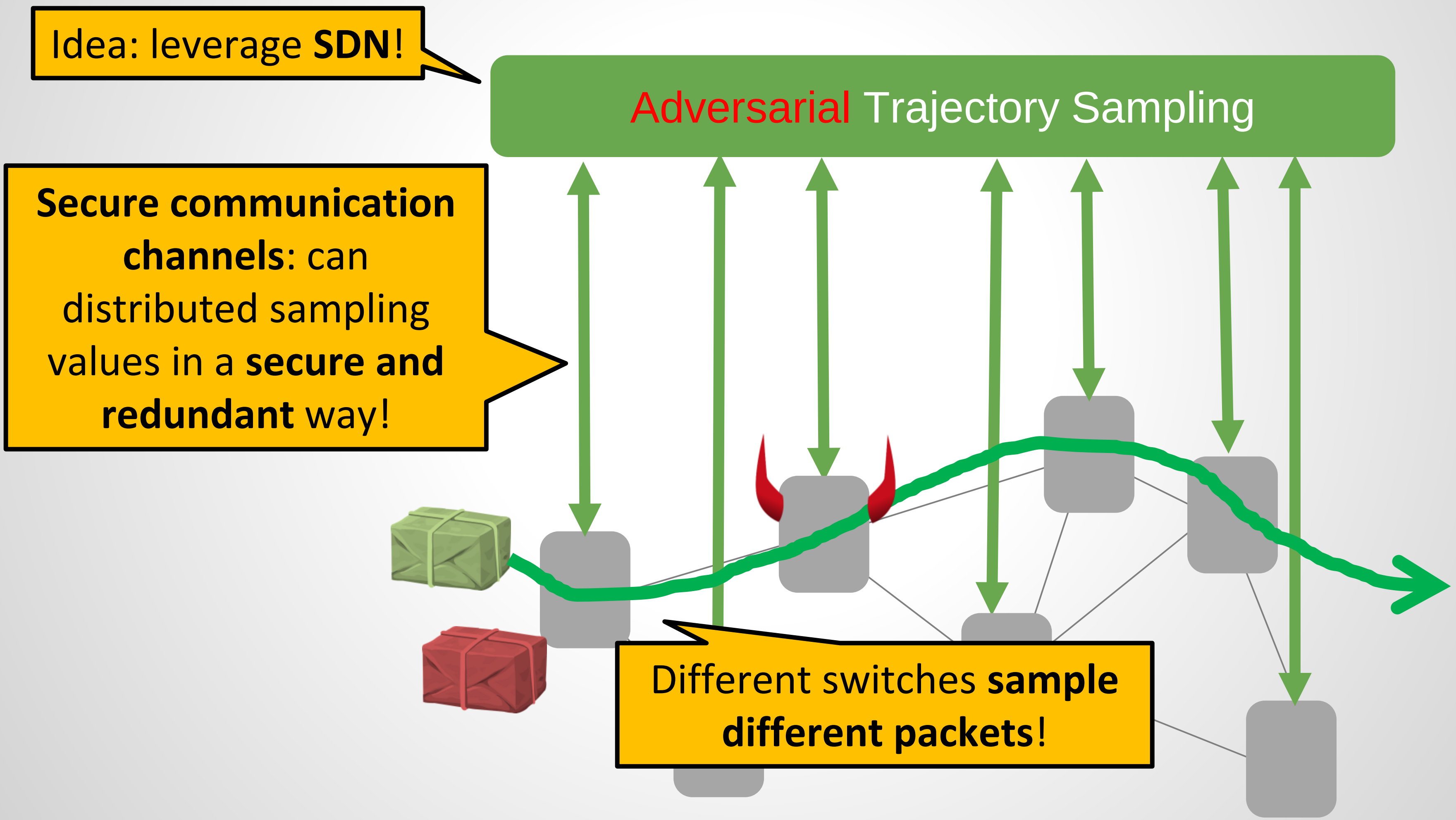
Idea: leverage **SDN**!



Secure communication channels: can distributed sampling values in a **secure and redundant** way!

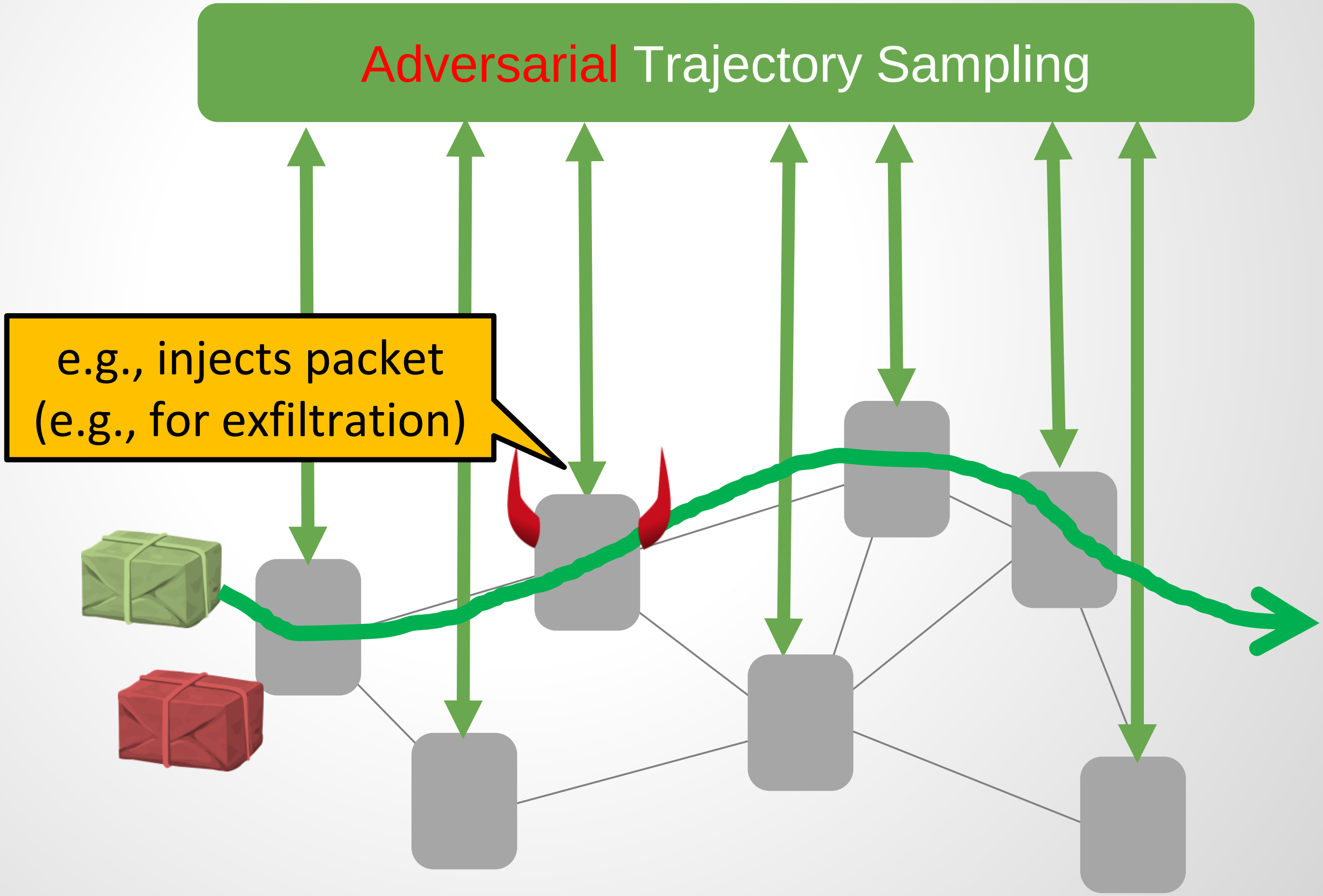
Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling



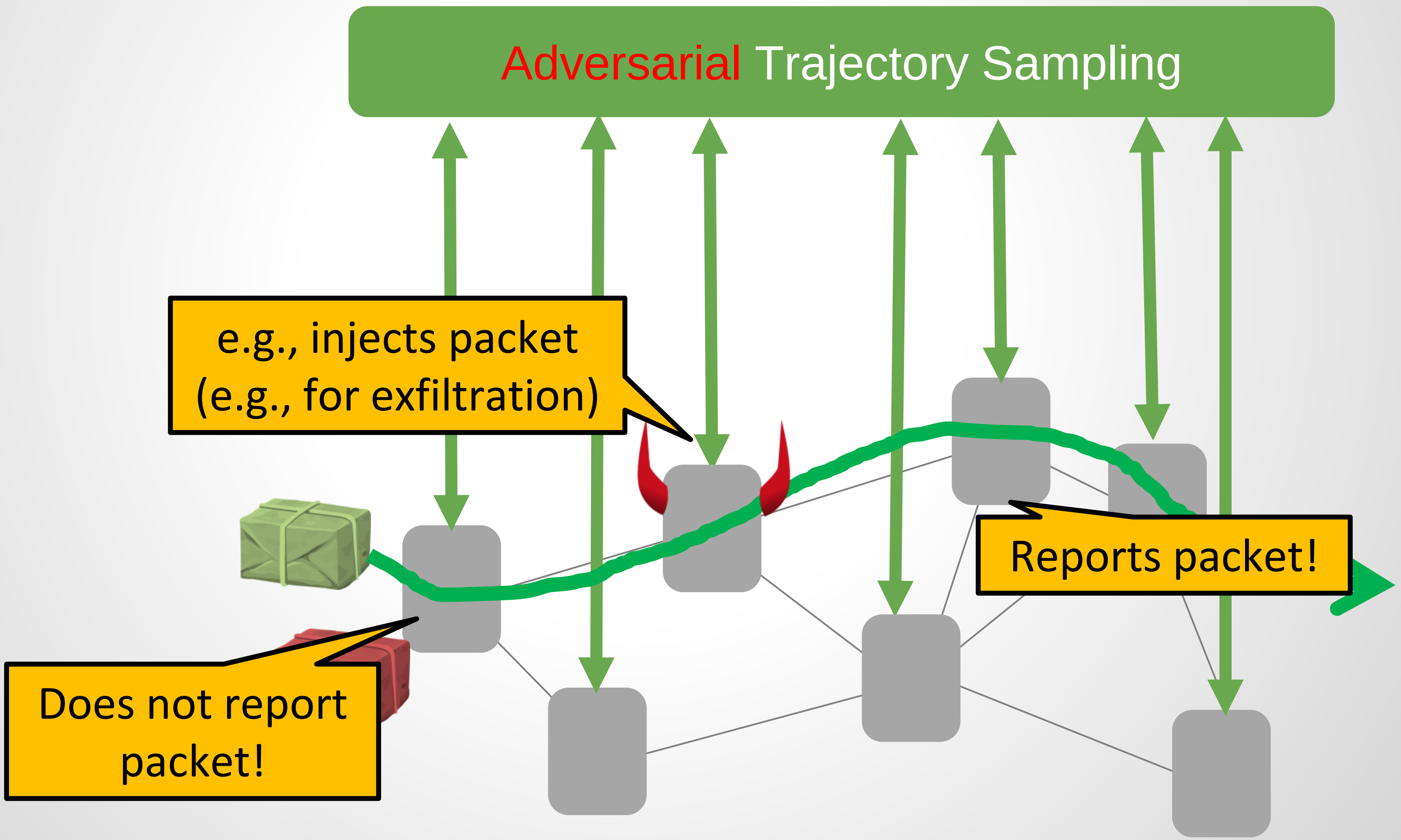
Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling



Opportunity: Adversarial Trajectory Sampling

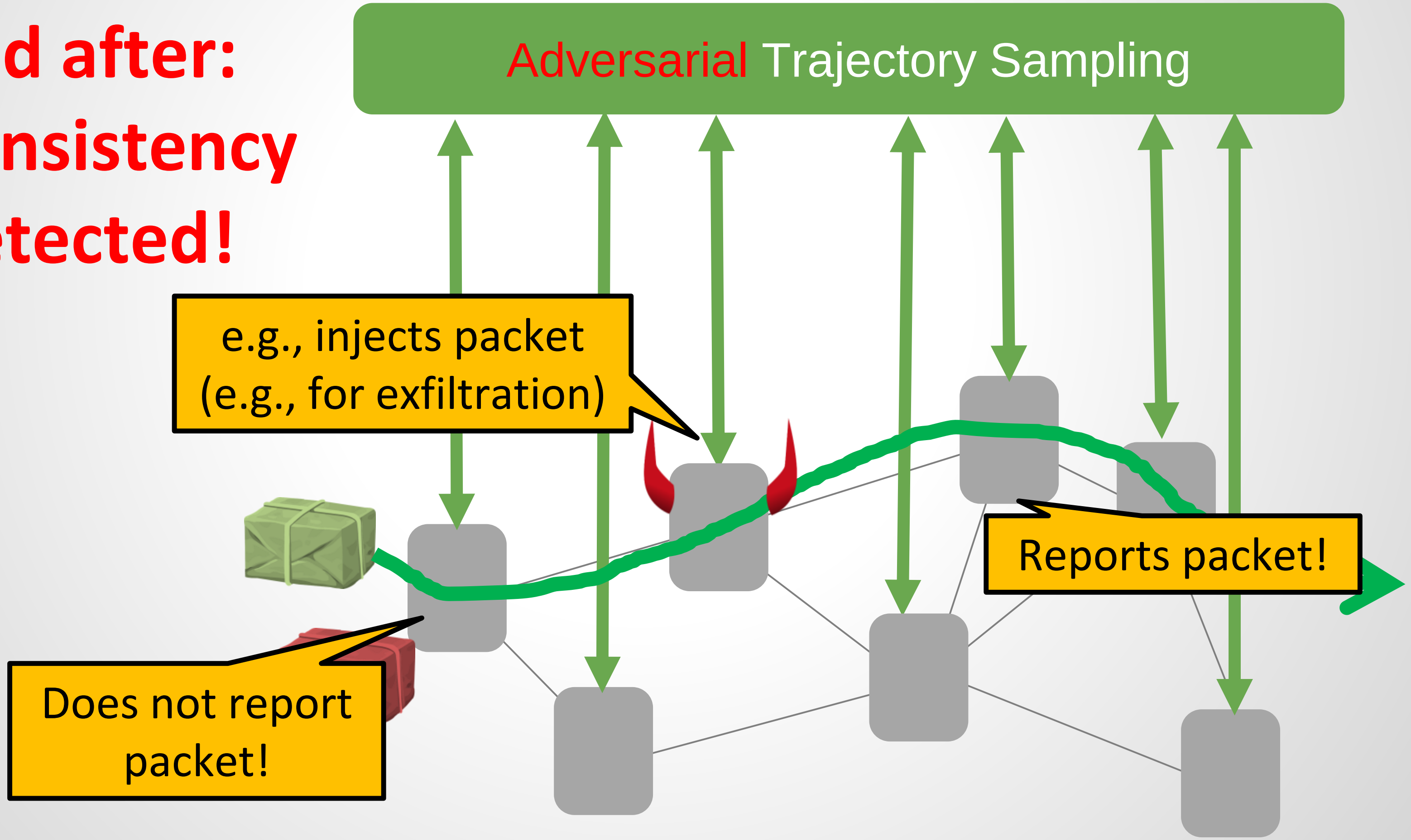
- Classic tool to monitor packet routes: trajectory sampling



Opportunity: Adversarial Trajectory Sampling

- Classic tool to monitor packet routes: trajectory sampling

If sampled before and after: Inconsistency detected!



Further Reading

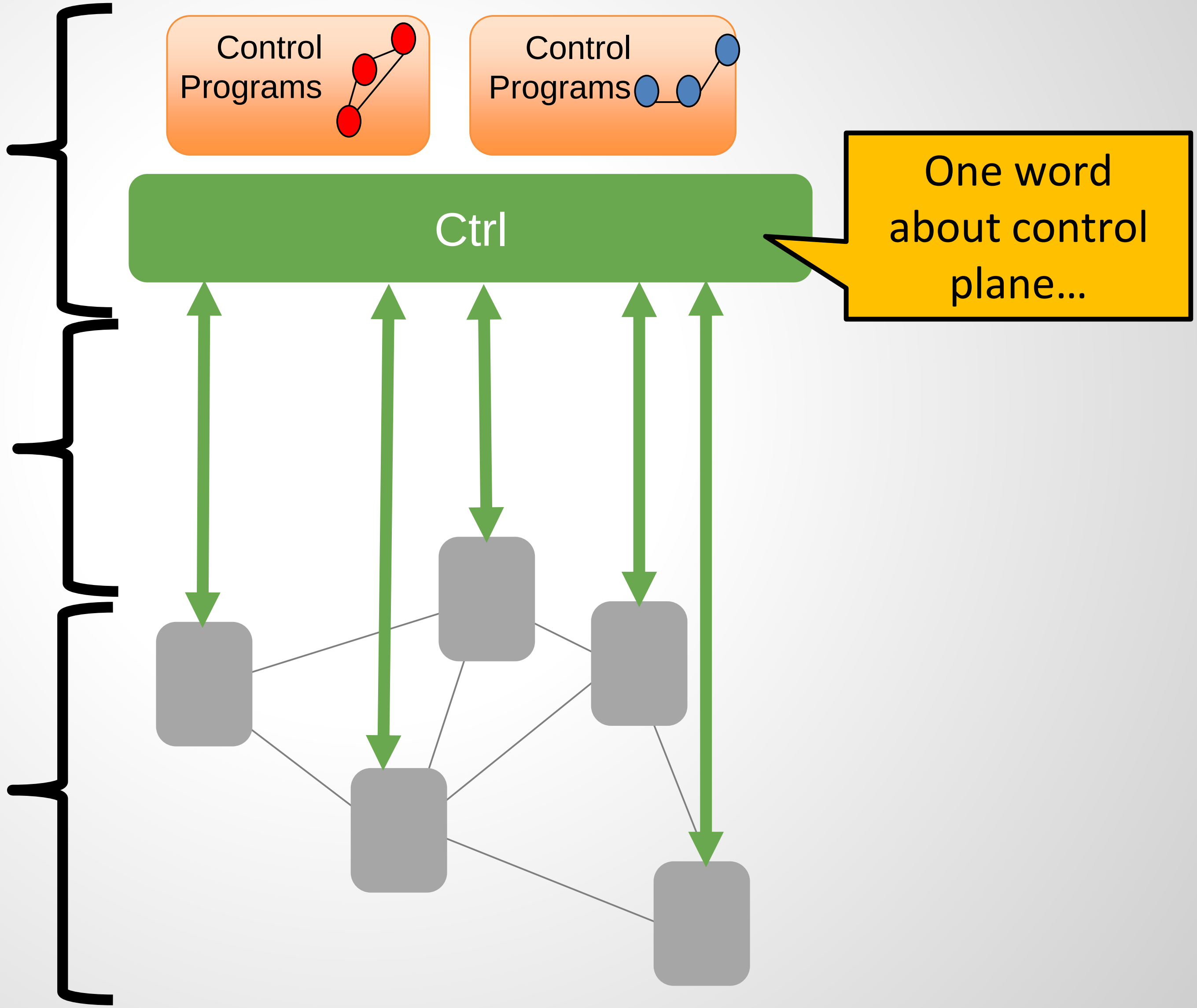
Software-Defined Adversarial Trajectory Sampling
Kashyap Thimmaraju, Liron Schiff, and Stefan Schmid.
ArXiv Technical Report, May 2017.

Algorithmic Problems in SDNs

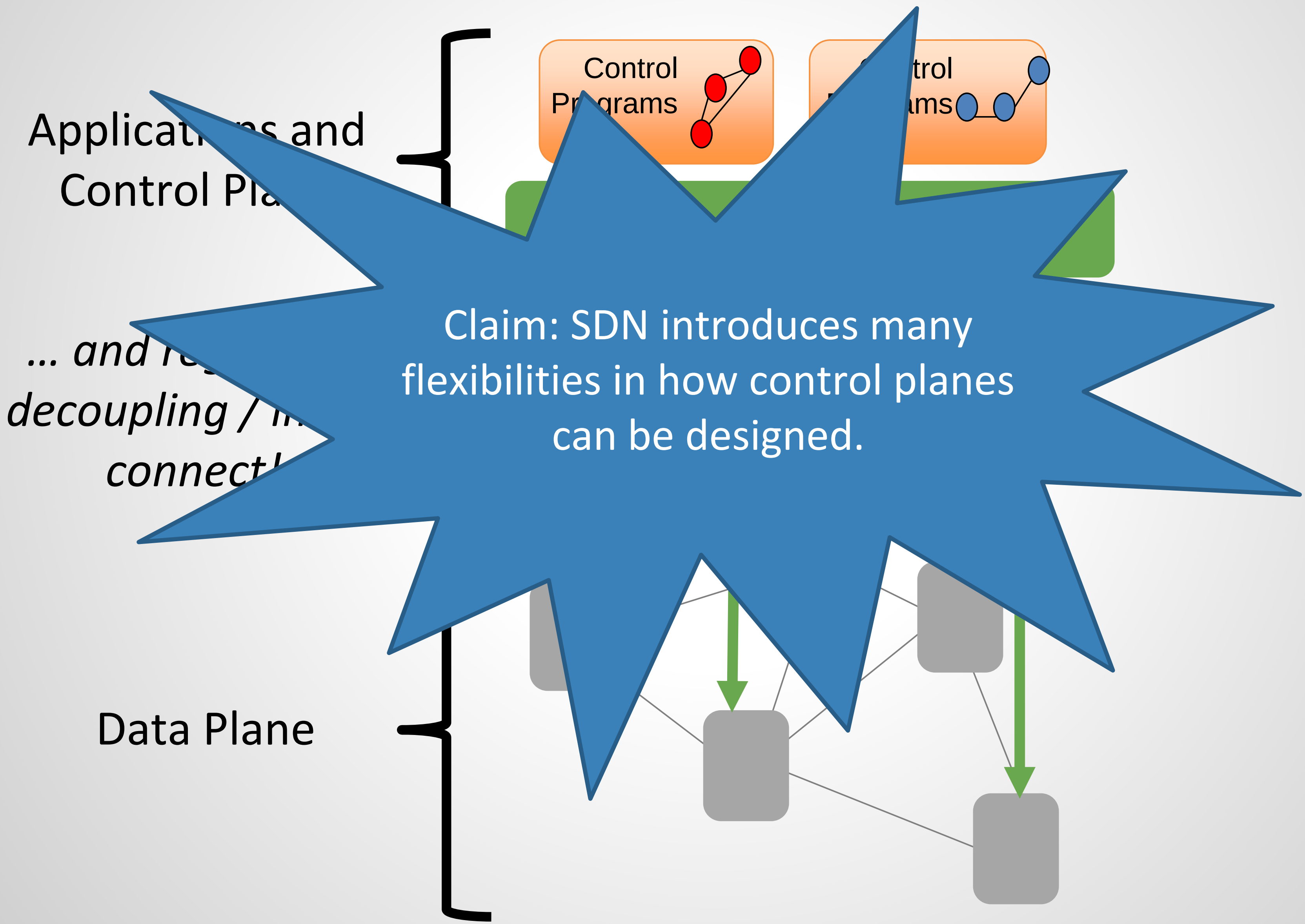
Applications and Control Plane

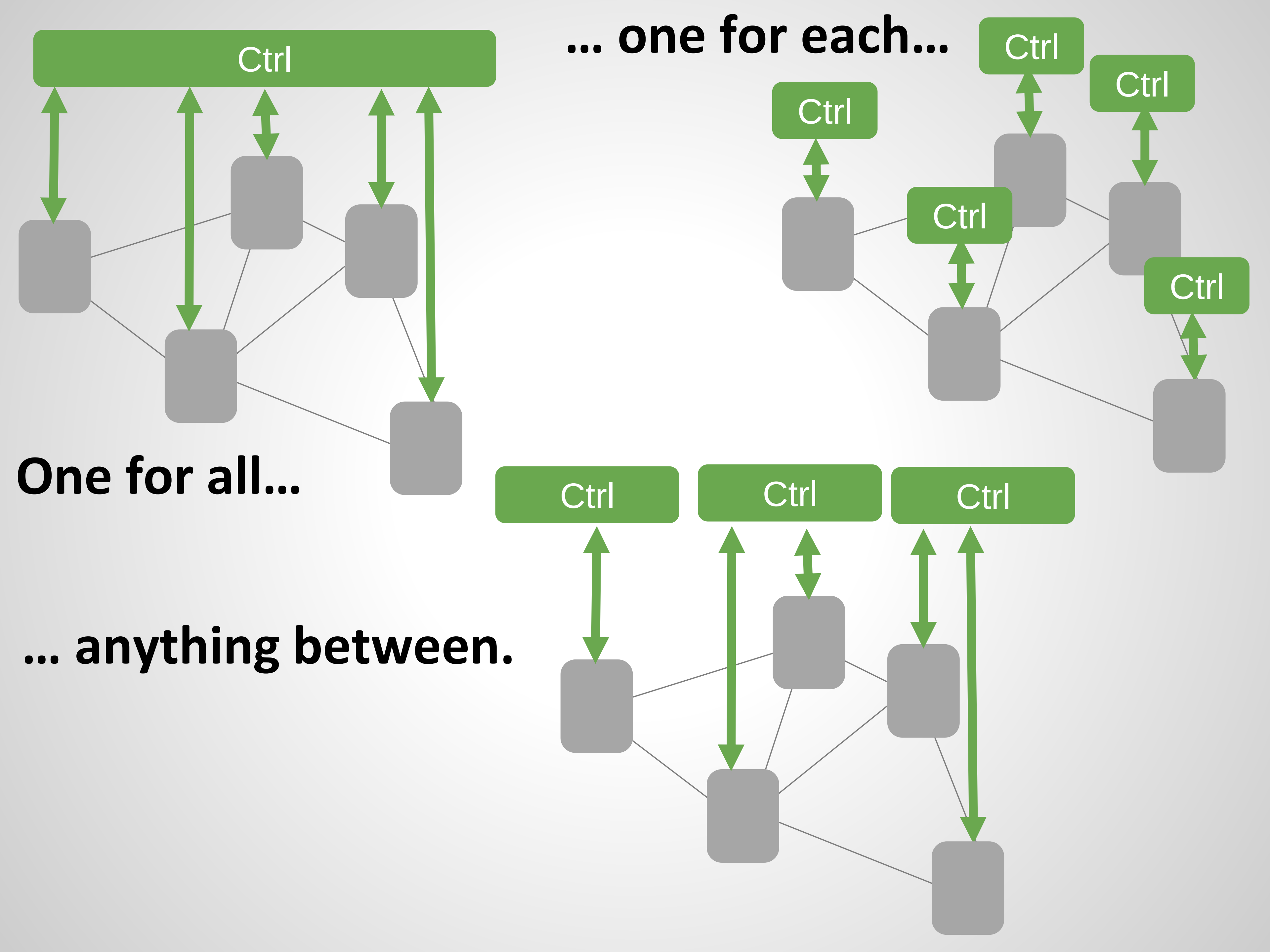
... and regarding decoupling / inter-connect!

Data Plane

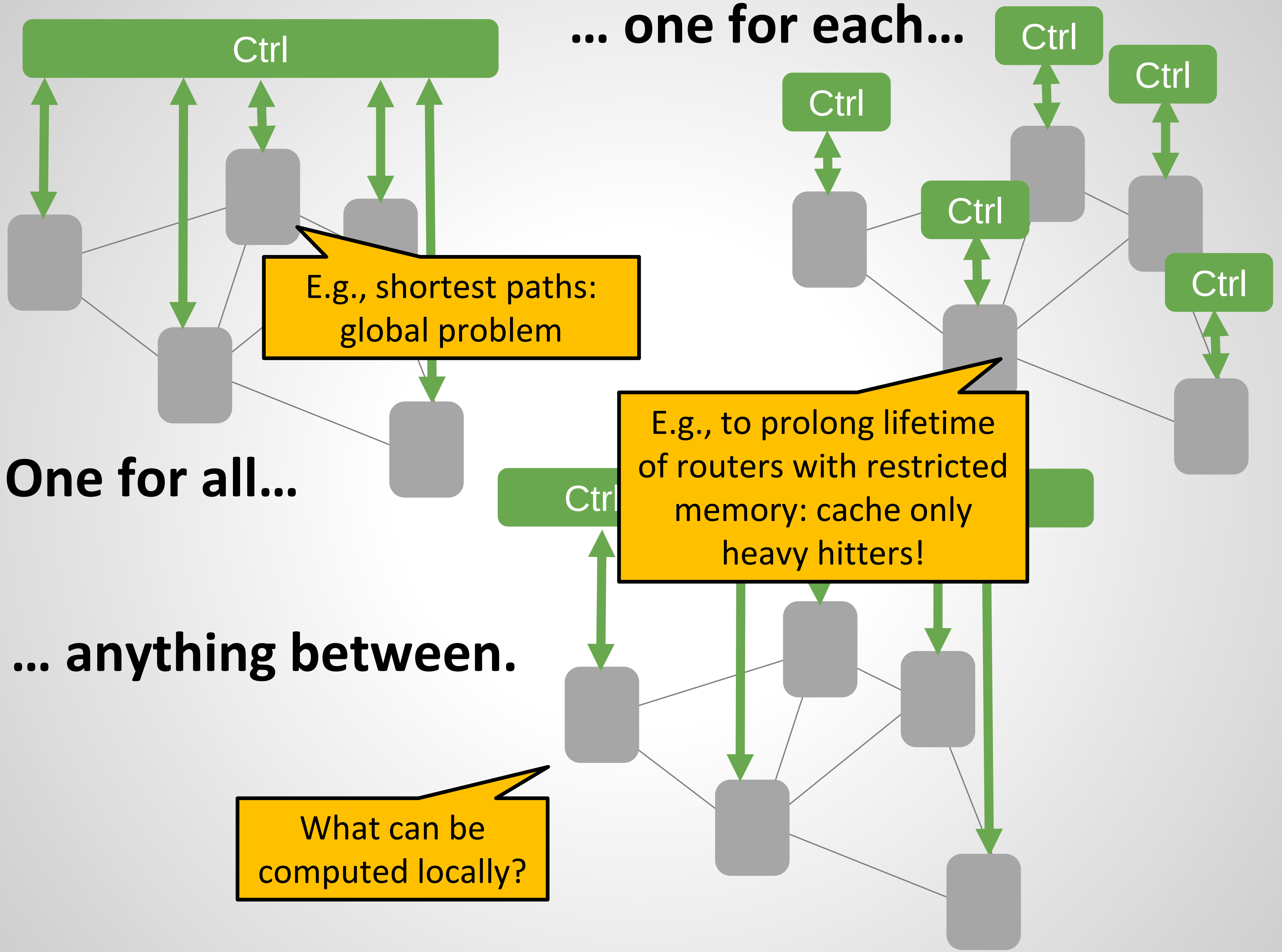


Algorithmic Problems in SDNs





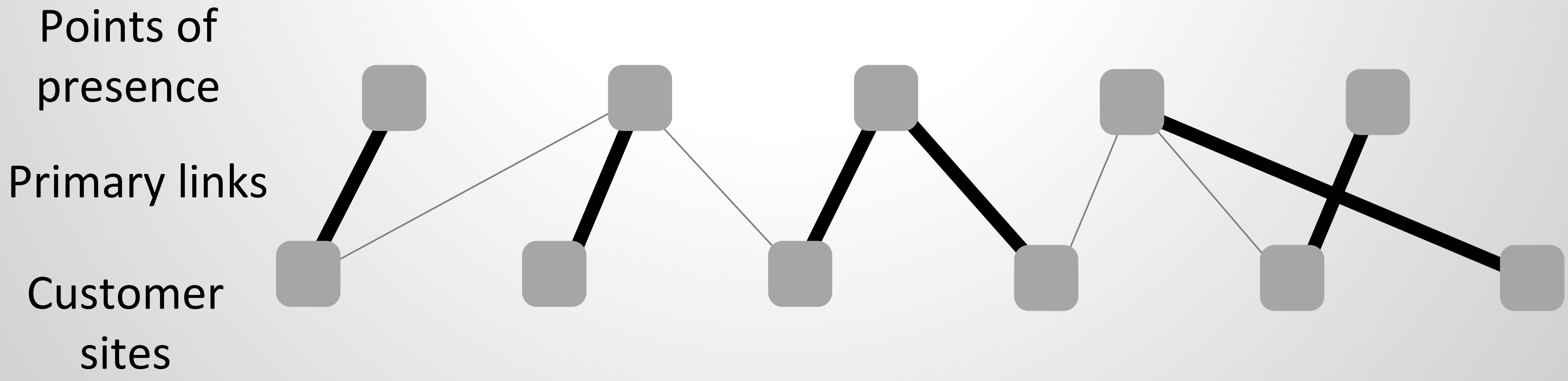
Ctrl
One for all...
... one for each...
Ctrl
Ctrl
Ctrl
Ctrl
Ctrl
... anything between.
Ctrl
Ctrl
Ctrl



One for all…
… one for each…
… anything between.
Ctrl
Ctrl
Ctrl
Ctrl
Ctrl
Ctrl
Ctrl
E.g., shortest paths: global problem
E.g., to prolong lifetime of routers with restricted memory: cache only heavy hitters!
What can be computed locally?

Challenge: Right Level of Locality?

- Some insights from distributed computing are handy: but come in a new light!
- But finding right level of locality is non-trivial: tradeoff between inter-controller communication and quality of solution
- E.g., in load balancing

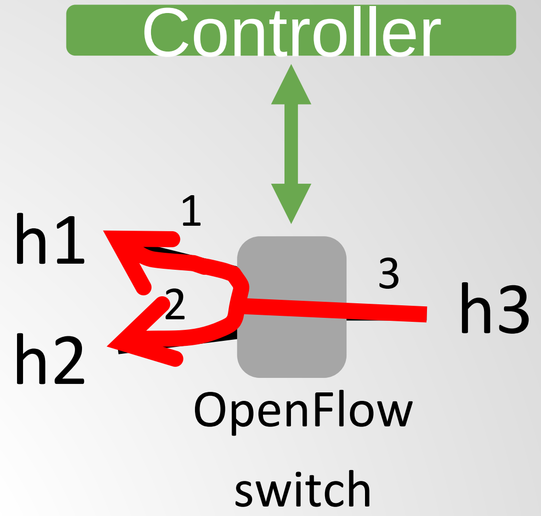


Further Reading

Exploiting Locality in Distributed SDN Control
Stefan Schmid and Jukka Suomela.
ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (**HotSDN**), Hong Kong, China, August 2013.

A Distributed and Robust SDN Control Plane for Transactional Network Updates
Marco Canini, Petr Kuznetsov, Dan Levin, and Stefan Schmid.
34th IEEE Conference on Computer Communications (**INFOCOM**), Hong Kong, April 2015.

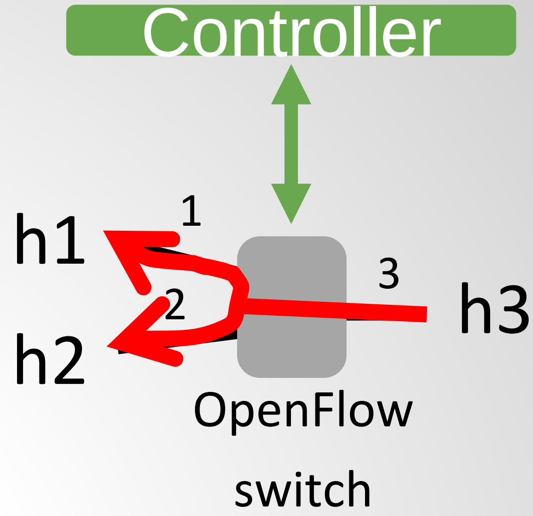
Jennifer Rexford's Example: SDN MAC Learning Done Wrong



- MAC learning: The «Hello World»
  - a bug in early controller versions
- In legacy networks simple
  - Flood packets sent to unknown destinations
  - Learn host's location when it sends packets (source address!)
- Pitfalls in SDN: learn sender => miss response
  - Assume: low priority rule * (no match): send to controller
  - h1->h2: Add rule h1@port1 (location learned)
  - Controller misses h2->h1 (as h1 known, h2 stay unknown!)
  - When h3->h2: flooding forever (learns h3, never learns h2)

Thanks to Jen Rexford for example!

Jennifer Rexford's Example: SDN MAC Learning Done Wrong

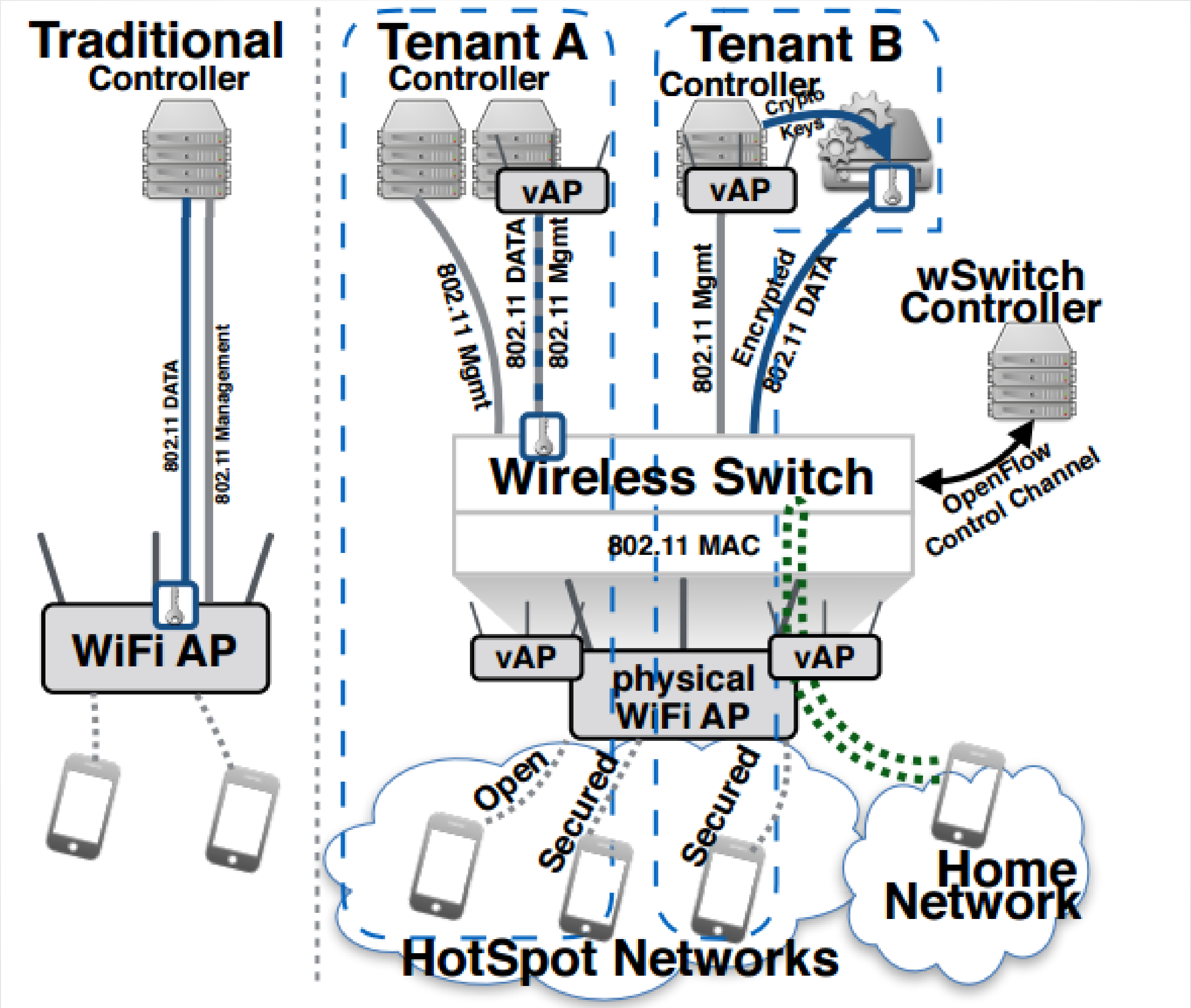


- MAC learning: The «Hello World»
  - a bug in early controller versions
- In legacy networks simple
  - Flood packets sent to unknown destinations
  - Learn host's location when it sends packets (source address!)
- Pitfalls in SDN: learn sender => miss response
  - Assume: low priority rule *
  - h1->h2: Add rule h1@port1
  - Controller misses h2->
  - When h3->h2: flooding forever (learns h3, never learns h2)

Controller never sees source h2: switch already knows all destinations h1 and h3, so for h2 it keeps flooding.

Thanks to Jen Rexford for example!

Software-Defined Wifi



Further Reading

OpenSDWN: Programmatic Control over Home and Enterprise WiFi
Julius Schulz-Zander, Carlos Mayer, Bogdan Ciobotaru, Stefan Schmid, and Anja Feldmann.
ACM Sigcomm Symposium on SDN Research (**SOSR**), Santa Clara, California, USA, June 2015.