SDN and Network Virtualization

Stefan Schmid

TU Berlin & Telekom Innovation Labs (T-Labs)

CleanSky ITN Summer School

Göttingen, Germany - September 14th-18th 2015

Rough Plan

- SDN and Network Virtualization: Debunking some myths and misunderstandings
- Some fundamental research challenges
- Mini-Tutorial: How to exploit flexiblities in virtualized networked systems?
- Mini-Tutorial: How are datacenters designed?
- Mini-Tutorial: Put your hands on SDN

Hands-On Exercises

- **In Groups:** you will get a VM and a key during coffee break

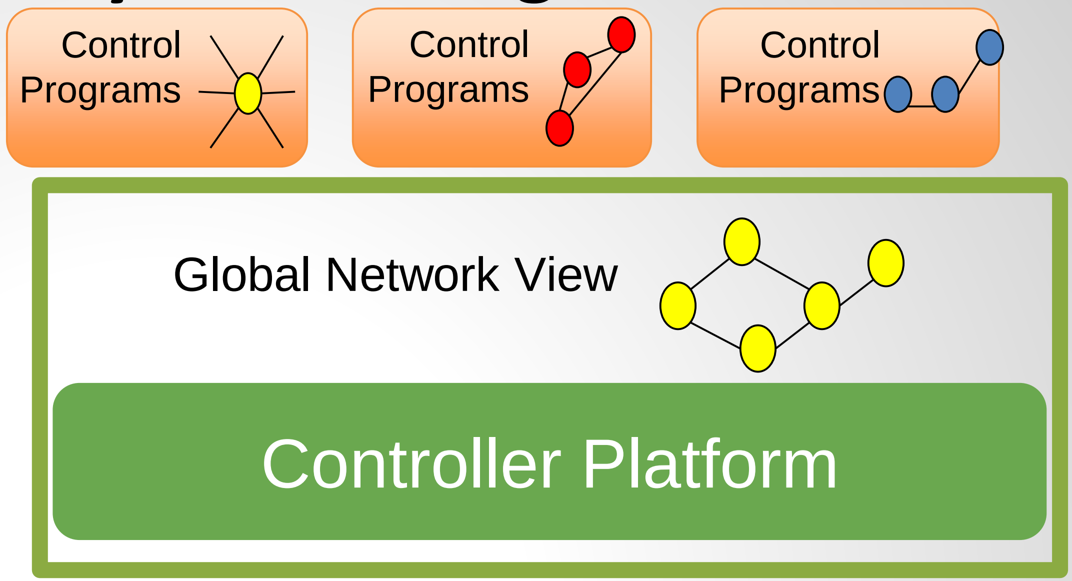
- **Exercise 1:**
  - Learning about waypoint enforcement and isolation concepts
  - Setting access control policies

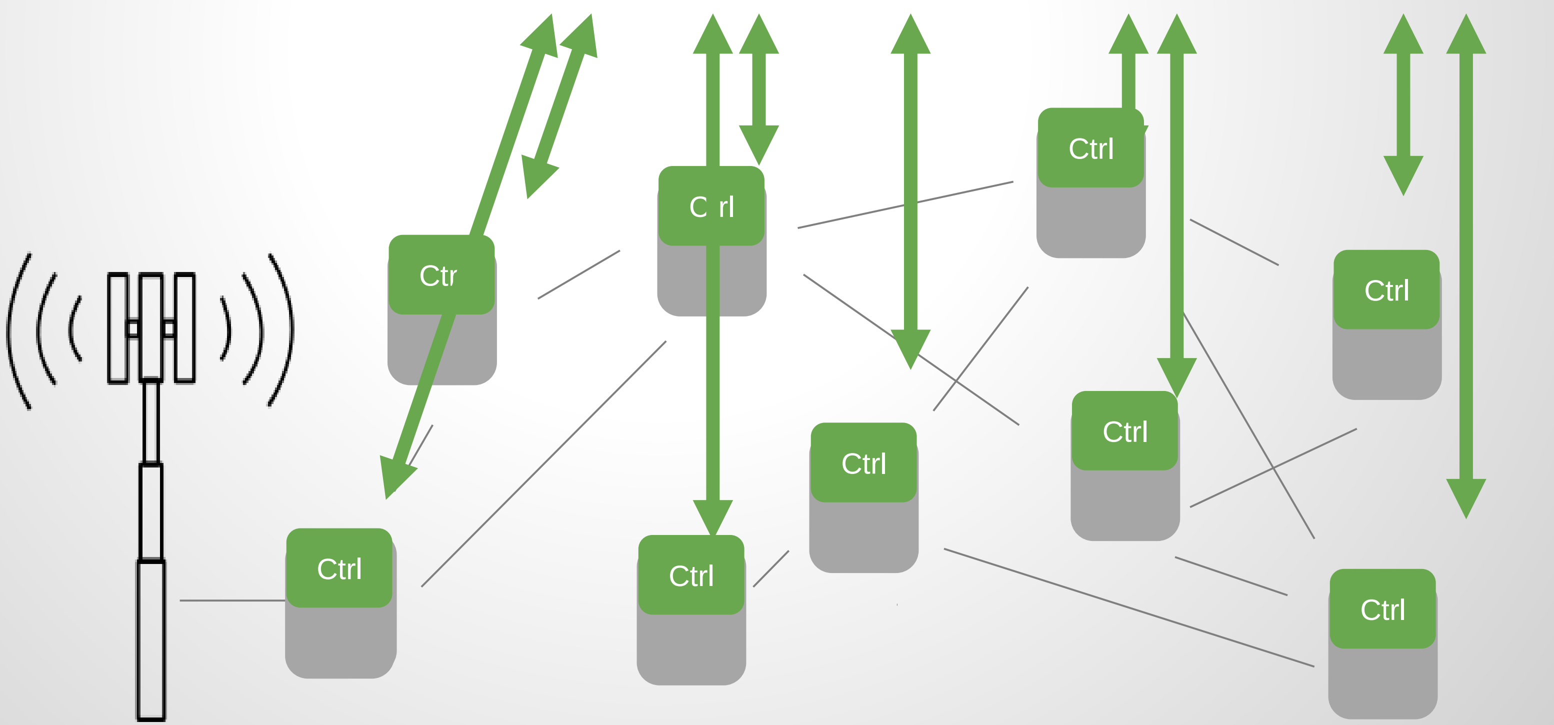
- **Exercise 2:**
  - Dealing with layer-3 devices (routers, WPE across subnet boundaries)
  - Supporting migration (IP subnet mobility)

- **Exercise 3:**
  - Dealing with layer-3 devices (routers)
  - Supporting migration

Flexible Networked Systems: Programmable…

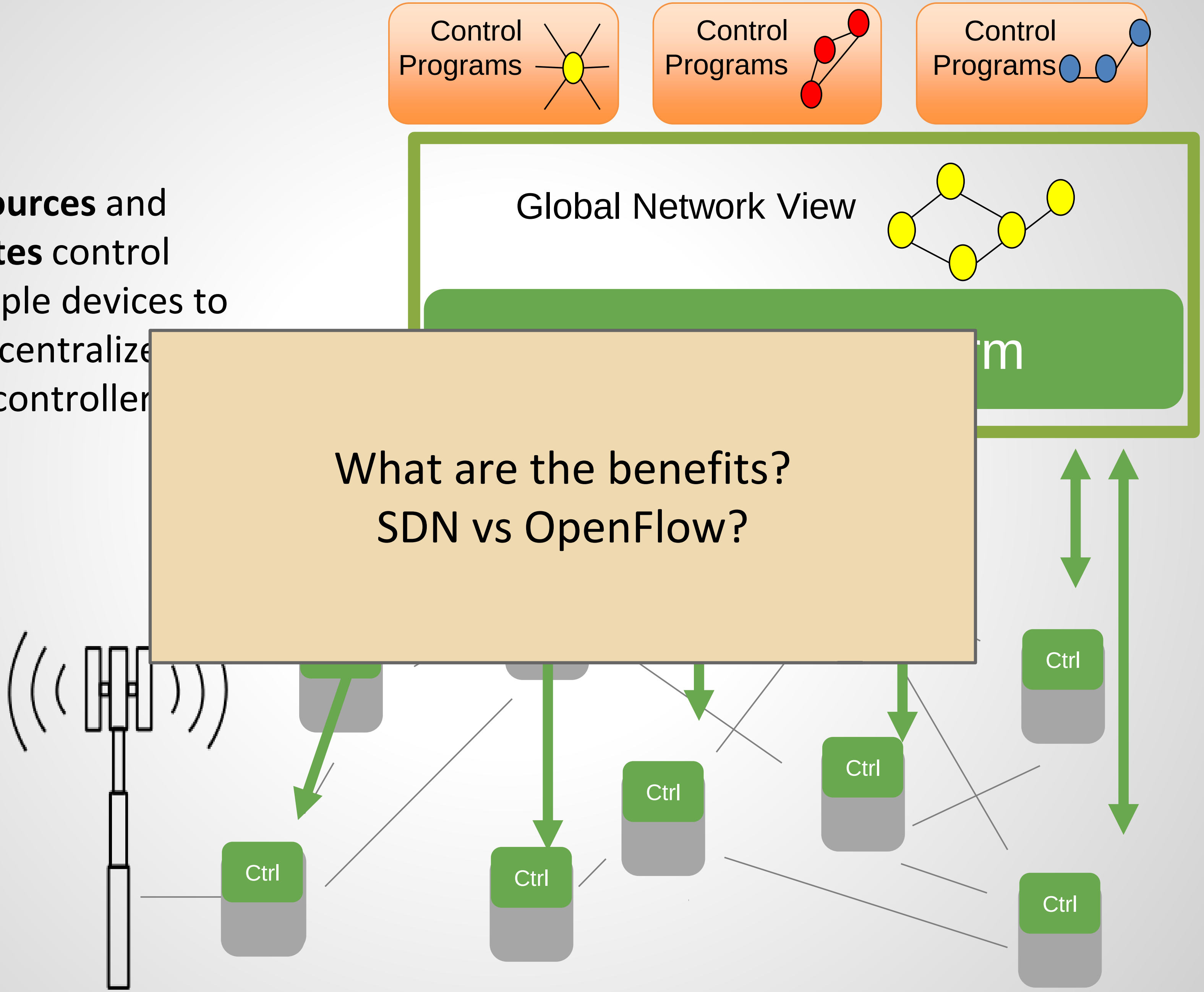
SDN **outsources** and **consolidates** control over multiple devices to (logically) centralized **software** controller





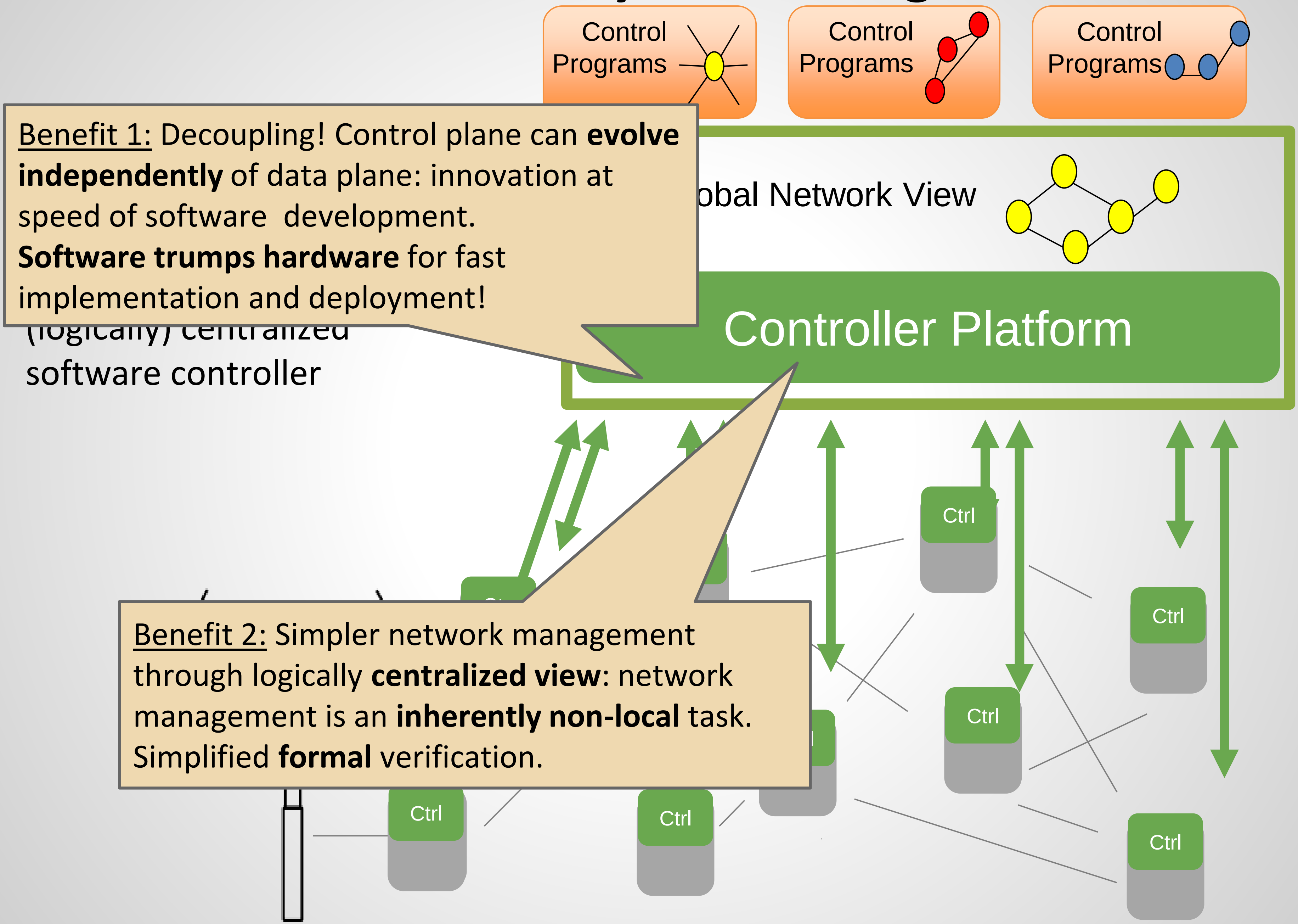
Flexible Networked Systems: Programmable...

SDN **outsources** and **consolidates** control over multiple devices to (logically) centralize[d] **software** controller[s]



What are the benefits?
SDN vs OpenFlow?

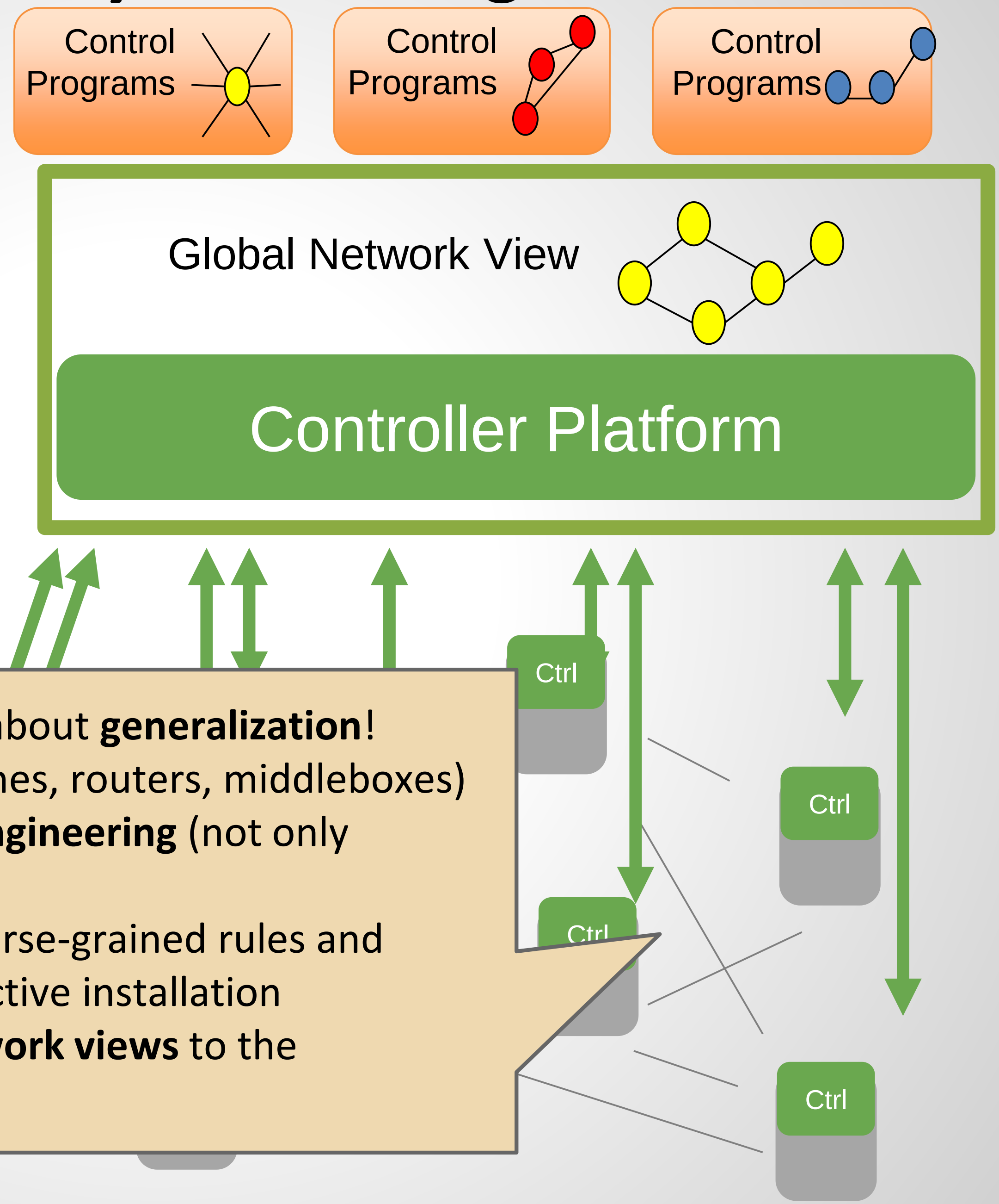
Flexible Networked Systems: Programmable...



Benefit 1: Decoupling! Control plane can **evolve independently** of data plane: innovation at speed of software development. **Software trumps hardware** for fast implementation and deployment!

Benefit 2: Simpler network management through logically **centralized view**: network management is an **inherently non-local** task. Simplified **formal** verification.

Flexible Networked Systems: Programmable...



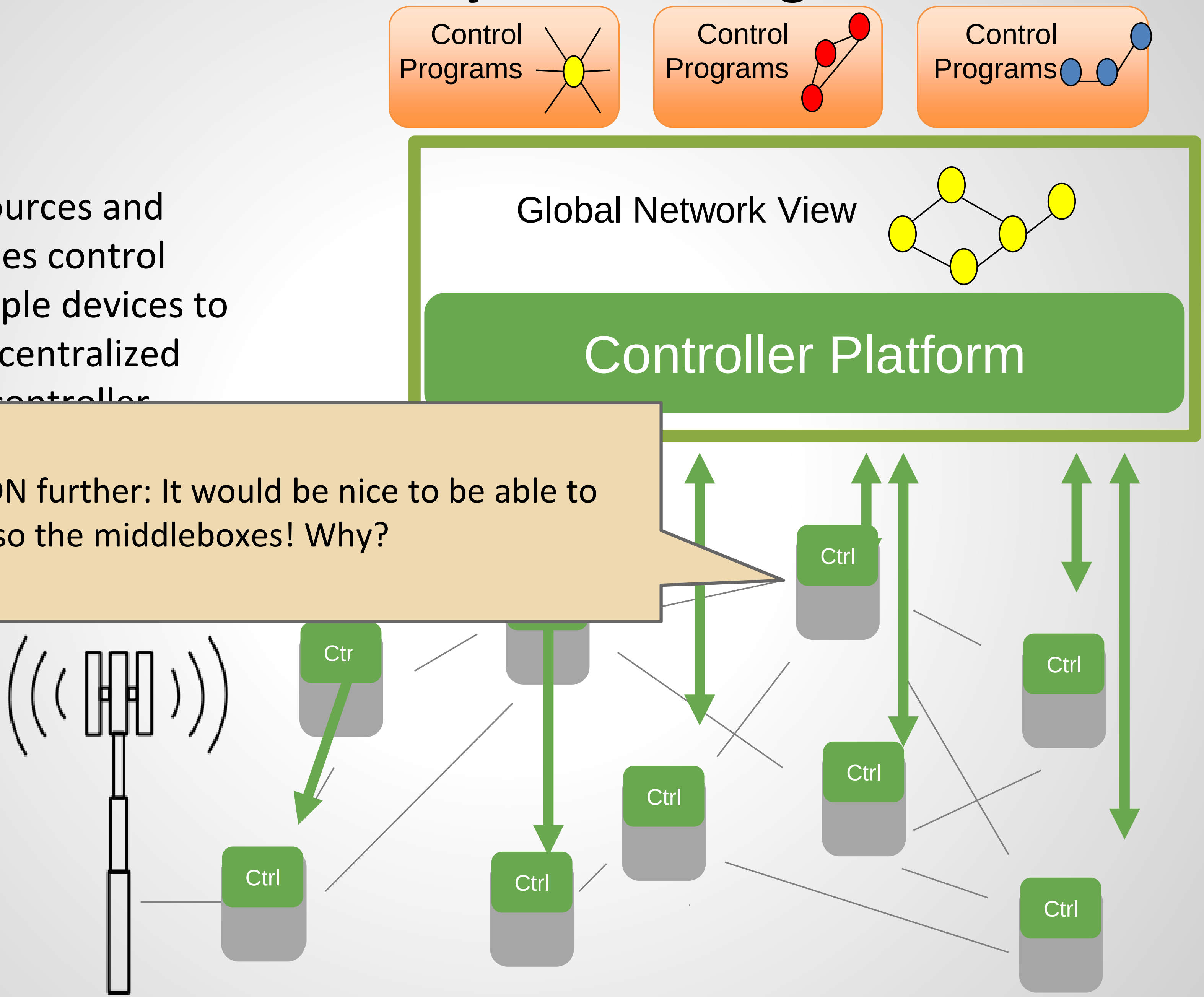
SDN outsources and consolidates control over multiple devices to (logically) centralized software controller

<u>Benefit 3:</u> Standard API OpenFlow is about **generalization**!

- Generalize **devices** (L2-L4: switches, routers, middleboxes)
- Generalize **routing and traffic engineering** (not only destination-based)
- Generalize **flow-installation**: coarse-grained rules and wildcards okay, proactive vs reactive installation
- Provide general and logical **network views** to the application / tenant

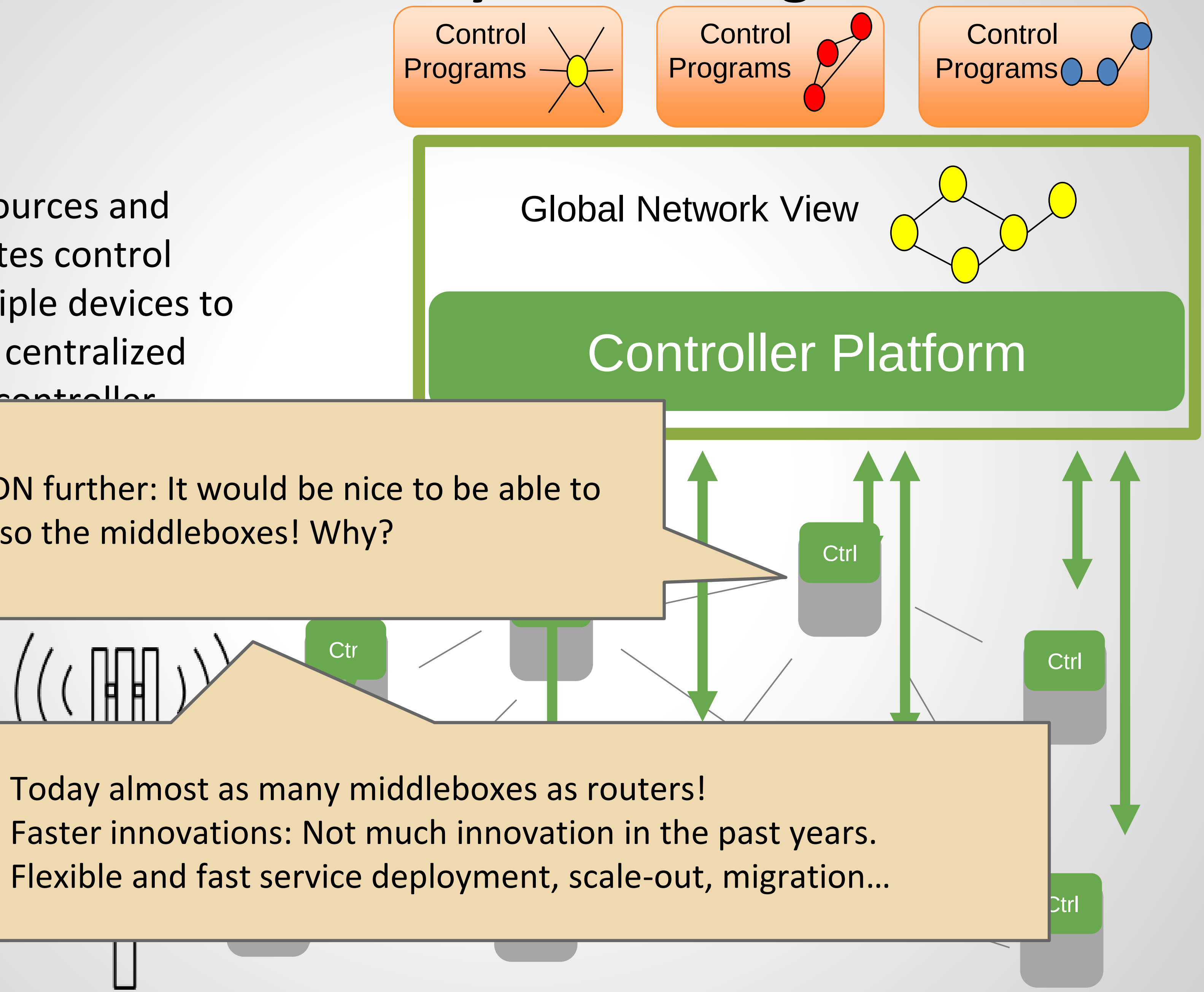
Flexible Networked Systems: Programmable...

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



Thinking SDN further: It would be nice to be able to program also the middleboxes! Why?

Flexible Networked Systems: Programmable...



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

Thinking SDN further: It would be nice to be able to program also the middleboxes! Why?

Today almost as many middleboxes as routers!
Faster innovations: Not much innovation in the past years.
Flexible and fast service deployment, scale-out, migration…

Flexible Networked Systems: ... *and Virtualized*

- Virtualization: a powerful concept in Computer Science

Flexible Networked Systems: ... *and Virtualized*

- Virtualization: a powerful concept in Computer Science

- Virtualization allows to **abstract**:
  - Hardware: compute, memory, storage, network resources
  - Or even entire distributed systems (including OS)

- **Decouples** the application from the substrate

- Introduces **flexibilities** for resource allocation
  - Improved **resource sharing** (esp. in commercial clouds)
  - Seamless migration

Challenges

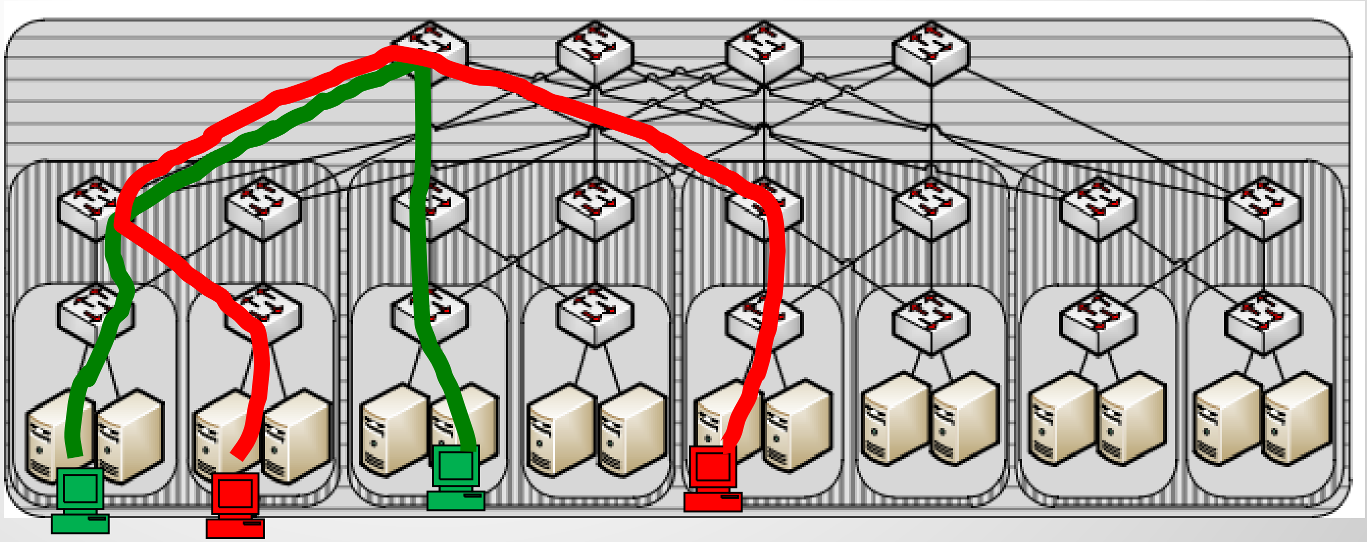
- Great…, but: are we brainstorming hammers or nails?
  - SDN and virtualization are enablers, ***not solutions***! What to do with them *and how*?

Challenges

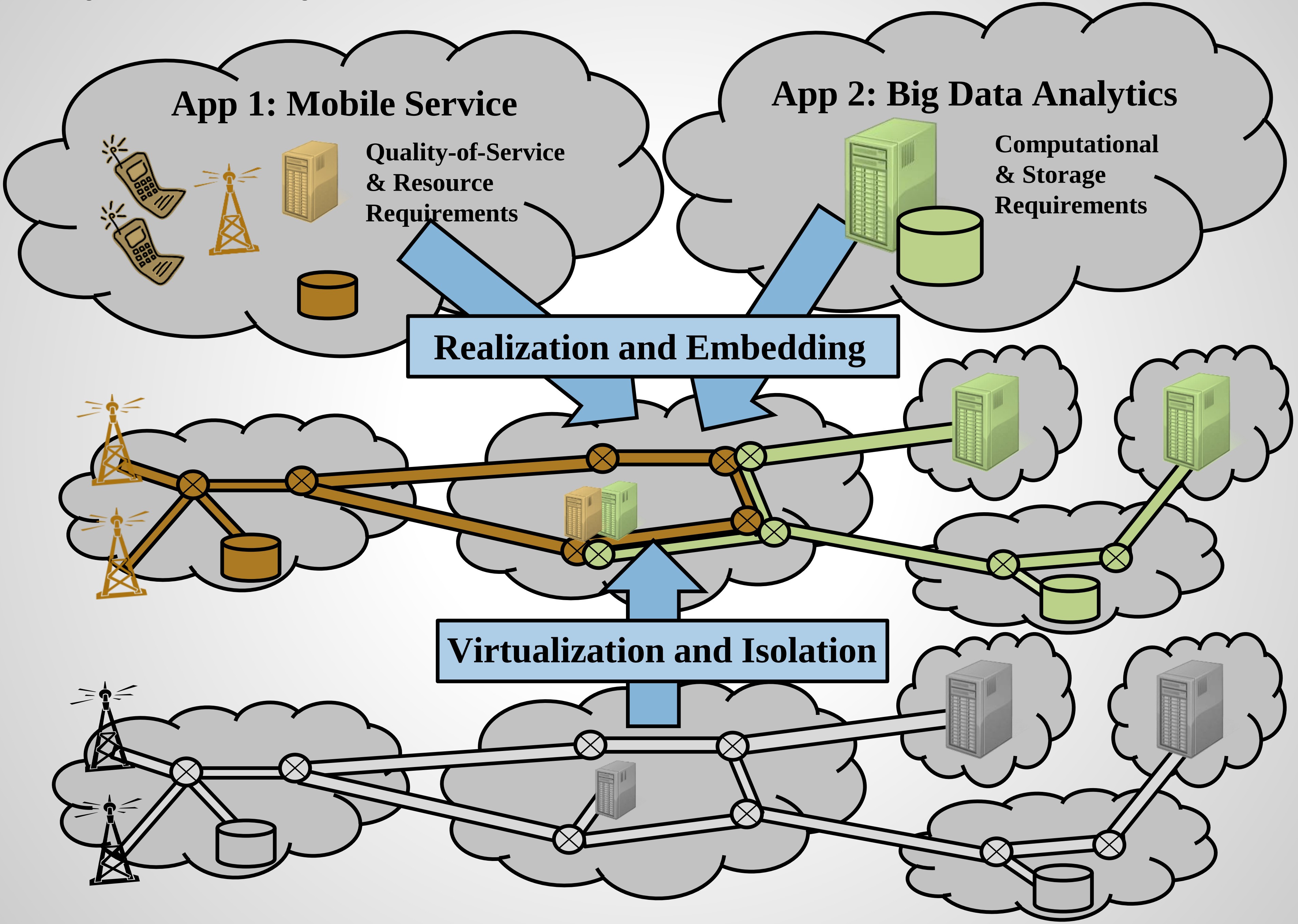
- Great…, but: are we brainstorming hammers or nails?
  - SDN and virtualization are enablers, ***not solutions***! What to do with them *and how*?

- Example: Virtualization for better **resource sharing**
  - Many **flexibilities** to embed **virtual machines**
  - But: often **not enough** to provide the expected performance!

Need to virtualize the **entire system**: otherwise risk of **interference** on other resources (network, CPU, memory, I/O) : **unredictable performance**



For predictable performance: full virtualization!



Network Virtualization

An interesting concept beyond datacenters…

- Network virtualization for the Internet!
- But what is the problem of the Internet today??

Network Virtualization

An interesting concept beyond datacenters…

- Network virtualization for the Internet!
- But what is the problem of the Internet today??
- Much innovation going on in the Internet
  - Peer-to-peer, online social networks, big data analytics, Internet-of-Things
  - But innovation limited to edge, no innovation in core! Example: IPv6
- Currently carrier networks are complex (VLAN, MPLS, …)
  - Based on blackboxes (CISCO routers, switches)
  - Consist of many middleboxes without uniform management
- Limited functionality
  - No path control

Network Virtualization

An interesting concept beyond datacenters…

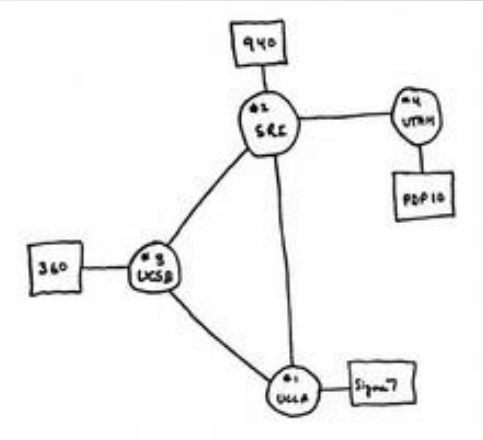
Paradigm to introduce innovation in Internet core

Indeed, Internet has changed radically over the last decades

Historic goal: Connectivity between a small set of super-computers

Applications: File transfer and emails among scientists

Situation now: Non-negligible fraction of the world population is constantly online



New requirements:

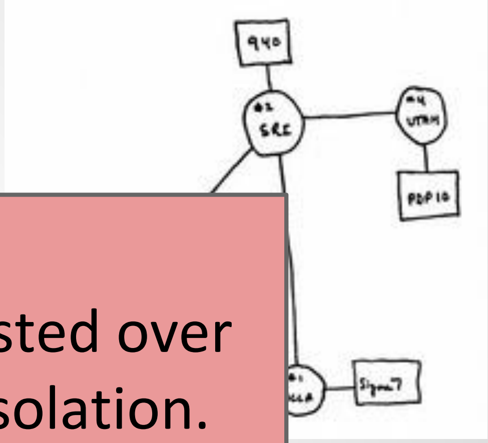
- More traffic, new demands on reliability and predictability
- Thus: use infrastructure more efficiently, use in-network caches: TE beyond destination-based routing, …
- Many different applications: Google docs vs datacenter synchronization vs on-demand video
- Also: user mobility, IP subnet mobility

Network Virtualization

An interesting concept beyond datacenters…

Paradigm to introduce innovation in Internet core

Indeed, Internet has changed radically over the last decades



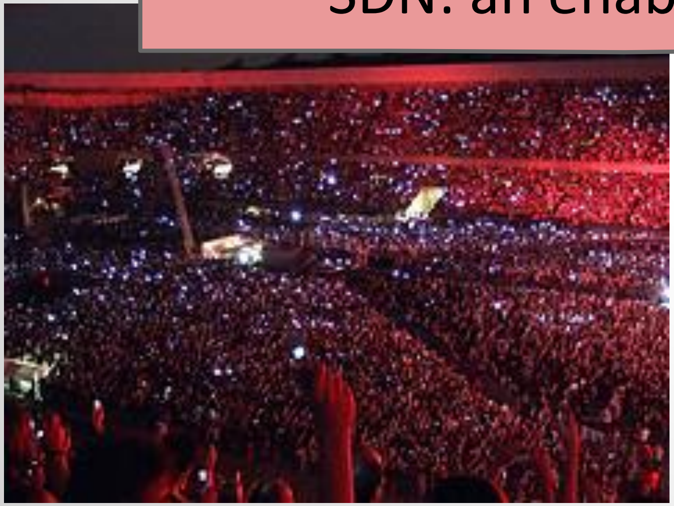
Histori

Applic

Situati
constar

Network Virtualization Vision:
Application tailored virtual networks, co-hosted over shared infrastructure, with (performance) isolation.

SDN: an enabler for network virtualization!



- More traffic, new demands on reliability and predictability
- Thus: use infrastructure more efficiently, use in-network caches: TE beyond destination-based routing, …
- Many different applications: Google docs vs datacenter synchronization vs on-demand video
- Also: user mobility, IP subnet mobility

 18

SDN and Virtualization: Many Algorithmic Challenges

- How to maximize the resource **utilization/sharing**?
  - E.g., how to embed a maximal number of virtual Hadoop clusters?
- And still ensure a **predictable** application performance?
  - How to **meet the job deadline** in MapReduce application?
  - How to guarantee **low lookup latencies** in data store?
  - It's not only about resource **contention**! **Skew** due to high demand also occurs in well-provisioned systems
- How to exploit allocation flexibilities to even mask and **compensate for** unpredictable events (e.g., failures)?
  - A key benefit of virtulization!

It's a Great Time to Be a Scientist

"We are at an interesting inflection point!"

Keynote by George Varghese at SIGCOMM 2014

Rough Plan

- **SDN and Network Virtualization: Debunking some myths and misunderstandings**

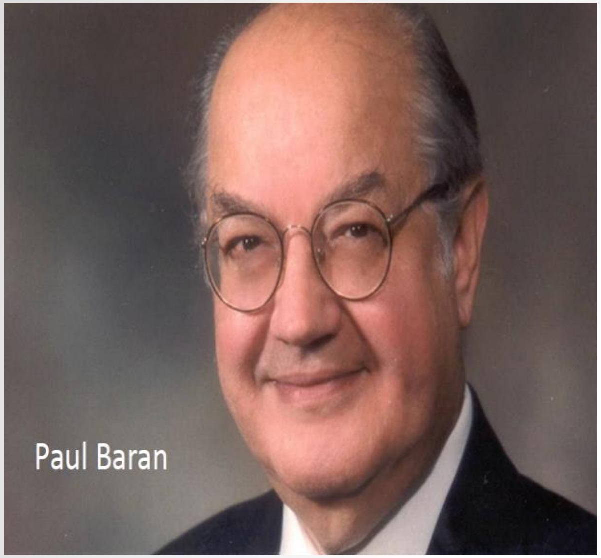
- Some fundamental research challenges

- Mini-Tutorial: How to exploit flexiblities in virtualized networked systems?

- Mini-Tutorial: How are datacenters designed?

- Mini-Tutorial: Put your hands on SDN

SDN: A step backward?



- Distributed control planes and packet-switching had a reason: "On Distributed Communications" (**Baran**, 1964)

- Packet-switched networks and distributed protocols render networks robust «to the bomb»

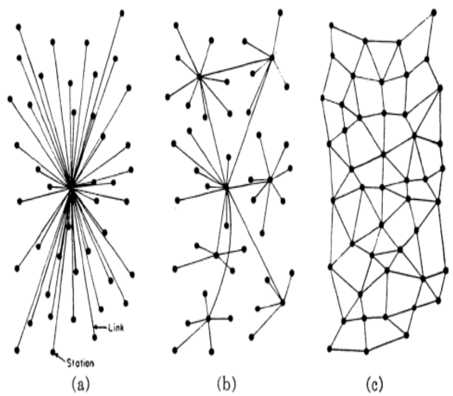


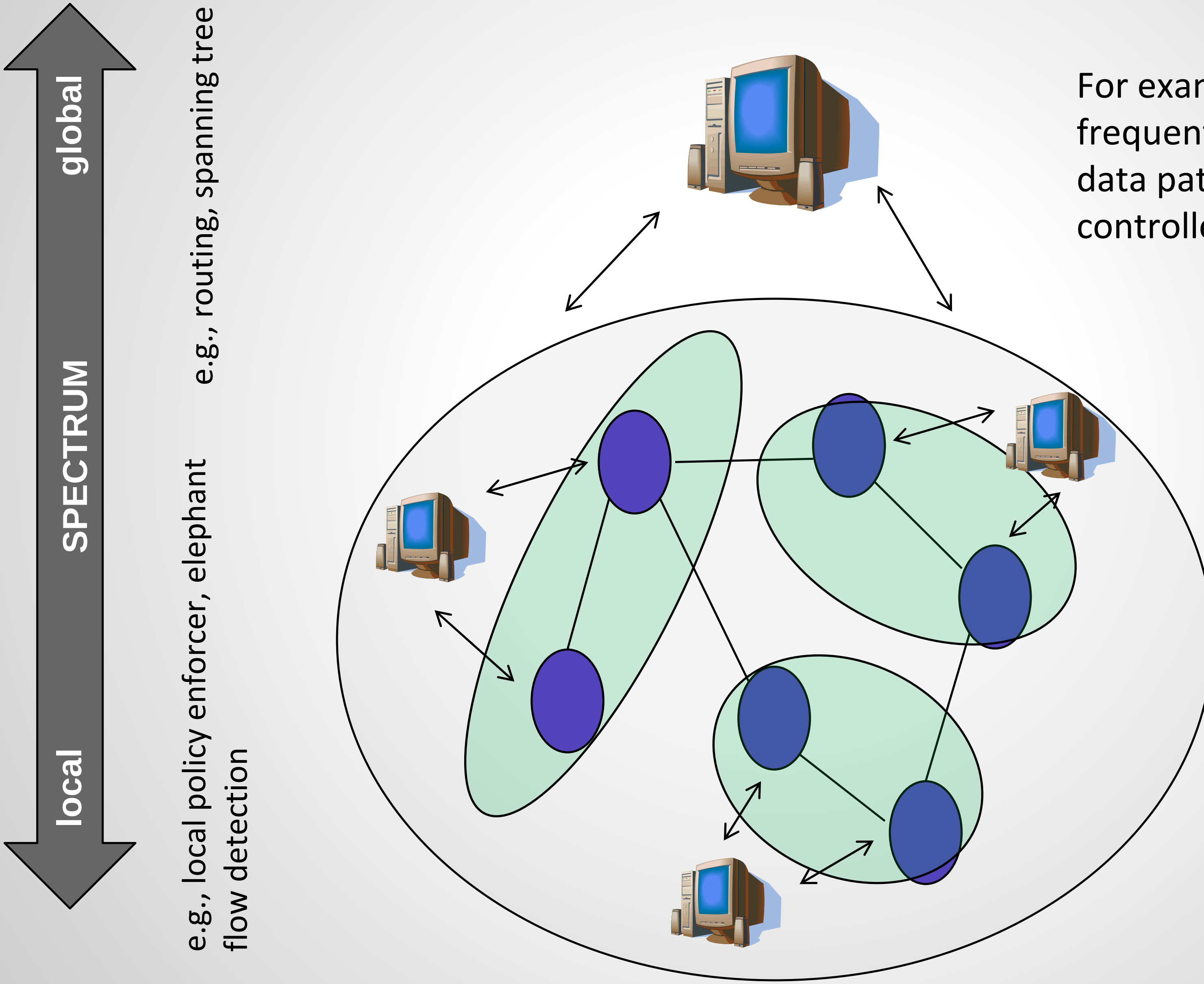
Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

1. Node & Edge Destruction
2. Distributed Routing

It's Logically Centralized!

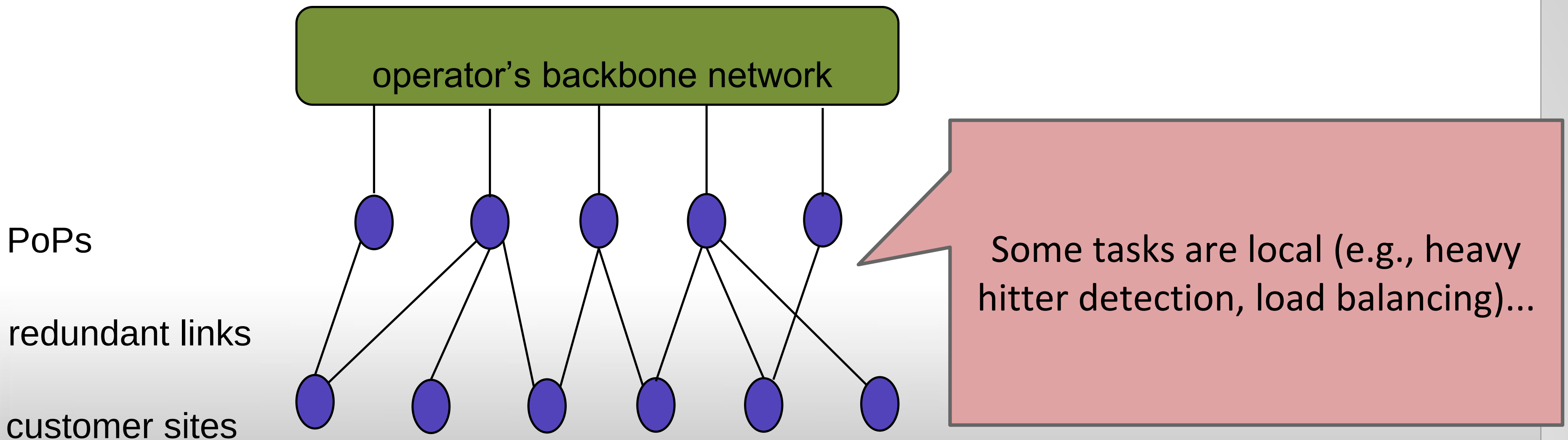
- SDN control only logically centralized: can actually be distributed!
  - Redundancy for reliability
  - Redundancy for elasticity (automatic scaling)
  - Geographic distribution: handle local events locally
- But building such distributed systems is not easy!

Challenge: Local Control

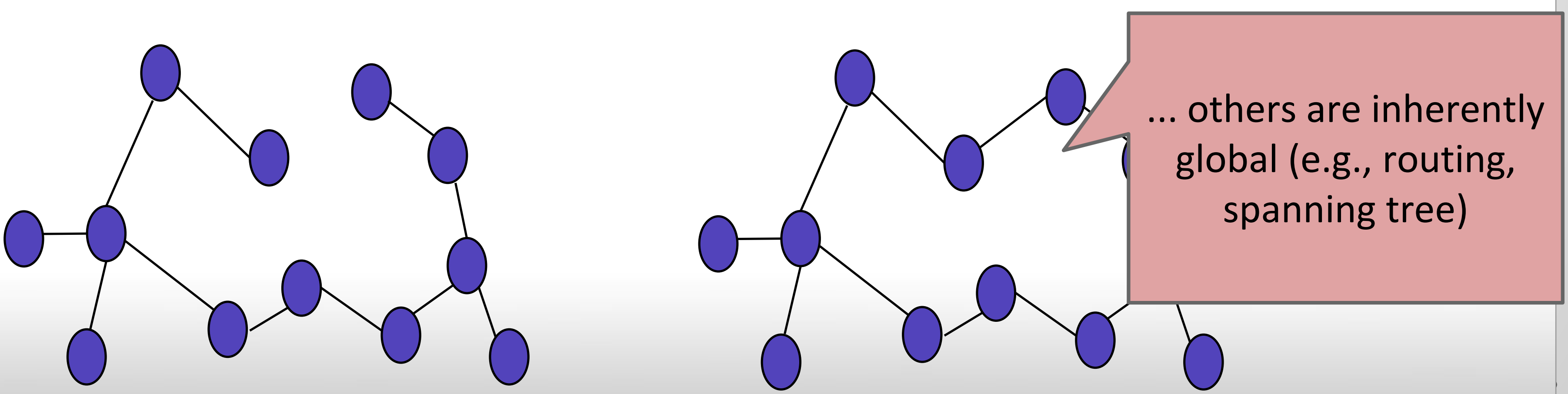


For example, handle frequent events close to data path, shield global controllers.

SDN Task 1: Link Assignment („Semi-Matching Problem“)

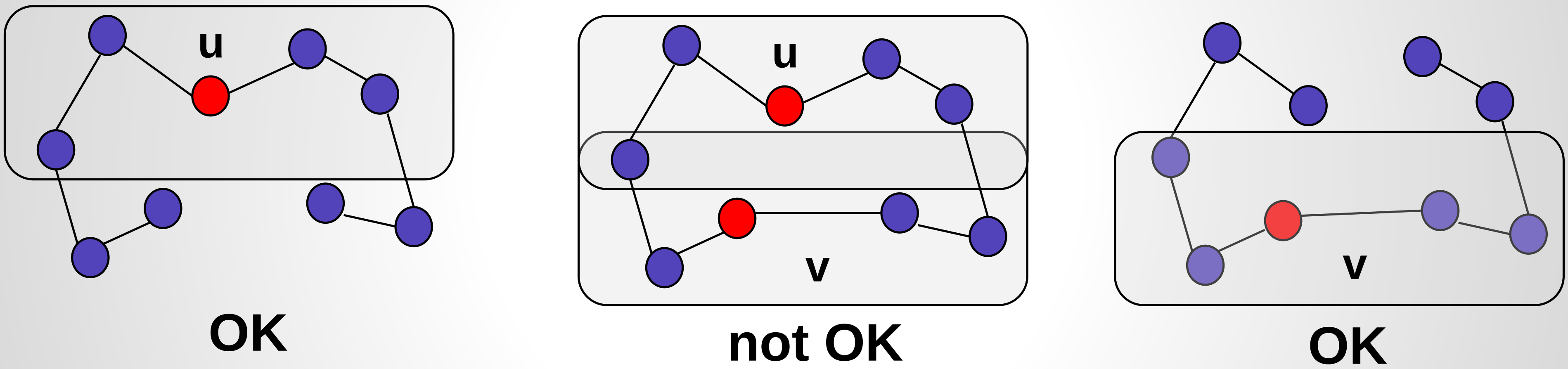


SDN Task 2: Spanning Tree Verification



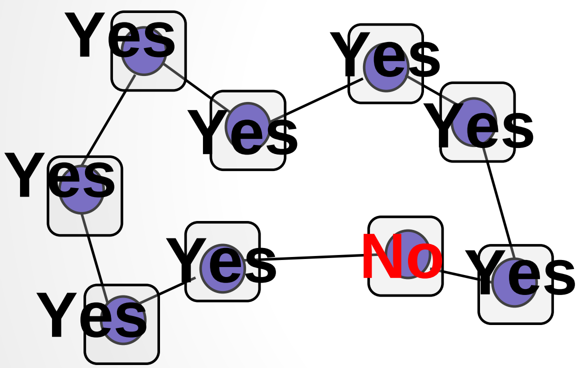
Limitations of a Local View

Example: checking loop-freedom



Verifying is easier!

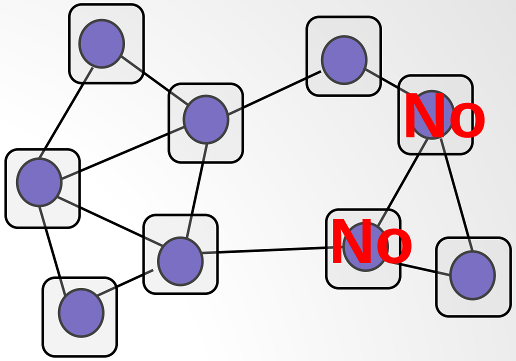
- Verification is easier than computation
  - Sometimes sufficient if at least one controller notices inconsistency: it can then trigger global re-computation

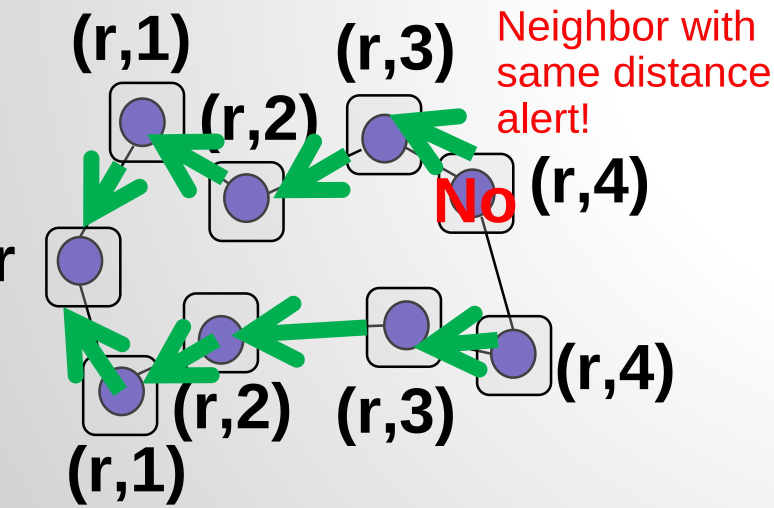
f( **) = No**

- Similar to classic computability theory
  - NP-complete problem solutions can be verified in polynomial time

Example

Euler Property: Hard to compute Euler tour ("each edge exactly once"), but easy to verify! 0-bits (= no communication) : output whether degree is even.



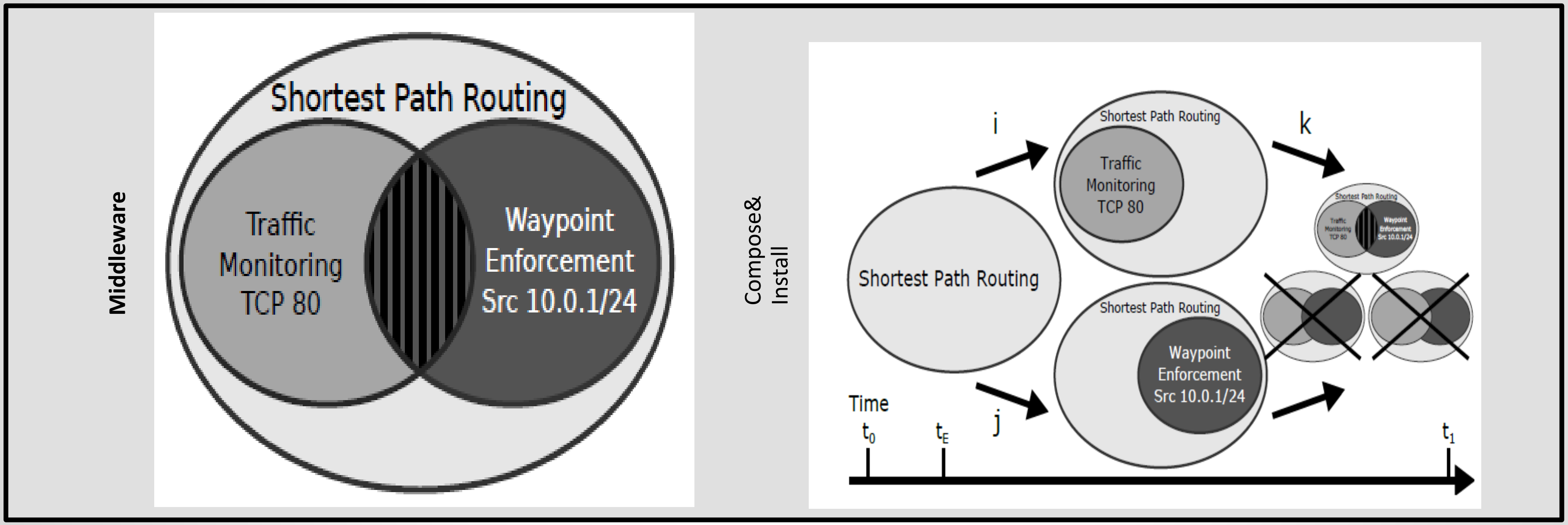


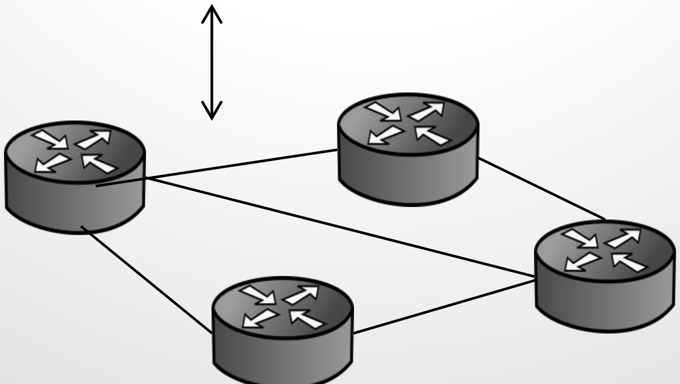
Spanning Tree Property: Label encodes root node plus distance & direction to root. At least one node notices that root/distance not consistent! Requires O(log n) bits.

How to deal with concurrency?

In charge of ACLs

In charge of tunnels

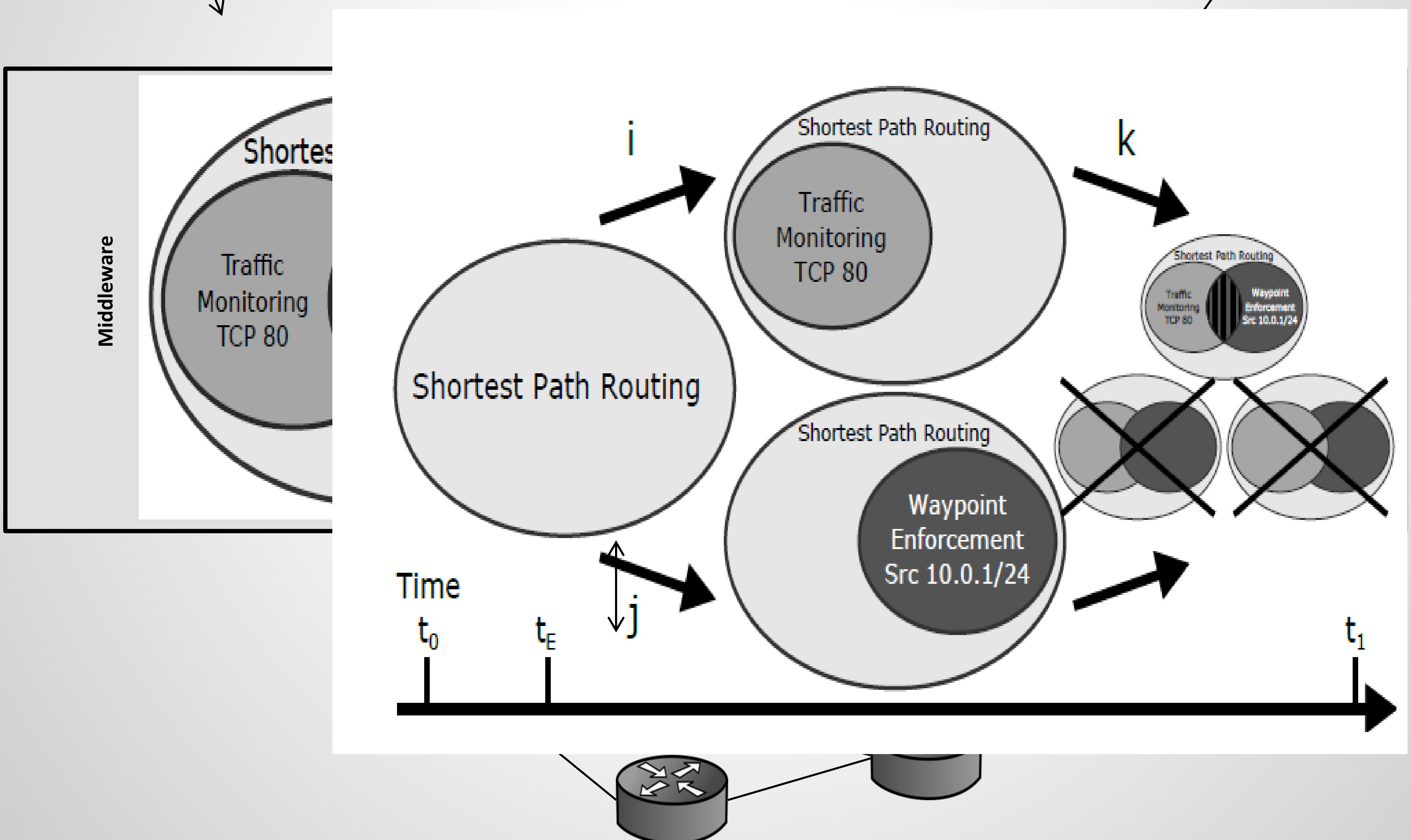




How to deal with concurrency?

In charge of ACLs

In charge of tunnels

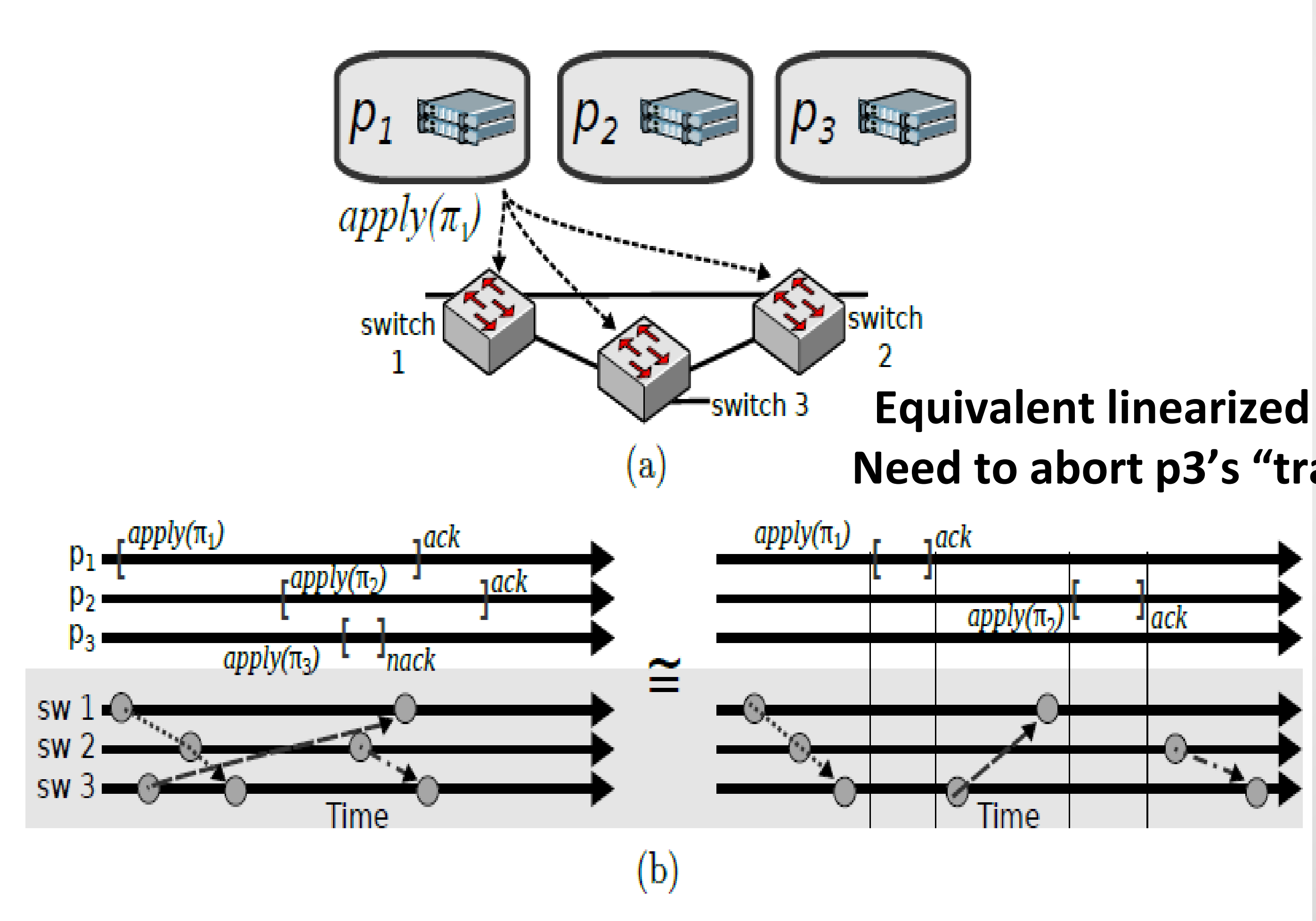


How to deal with concurrency?

A distributed systems problem!

Problem: Conflict free, per-packet consistent policy composition and installation

Holy Grails: Linearizability (Safety), Wait-freedom (Liveness)



Equivalent linearized schedule!
Need to abort p3's "transaction".

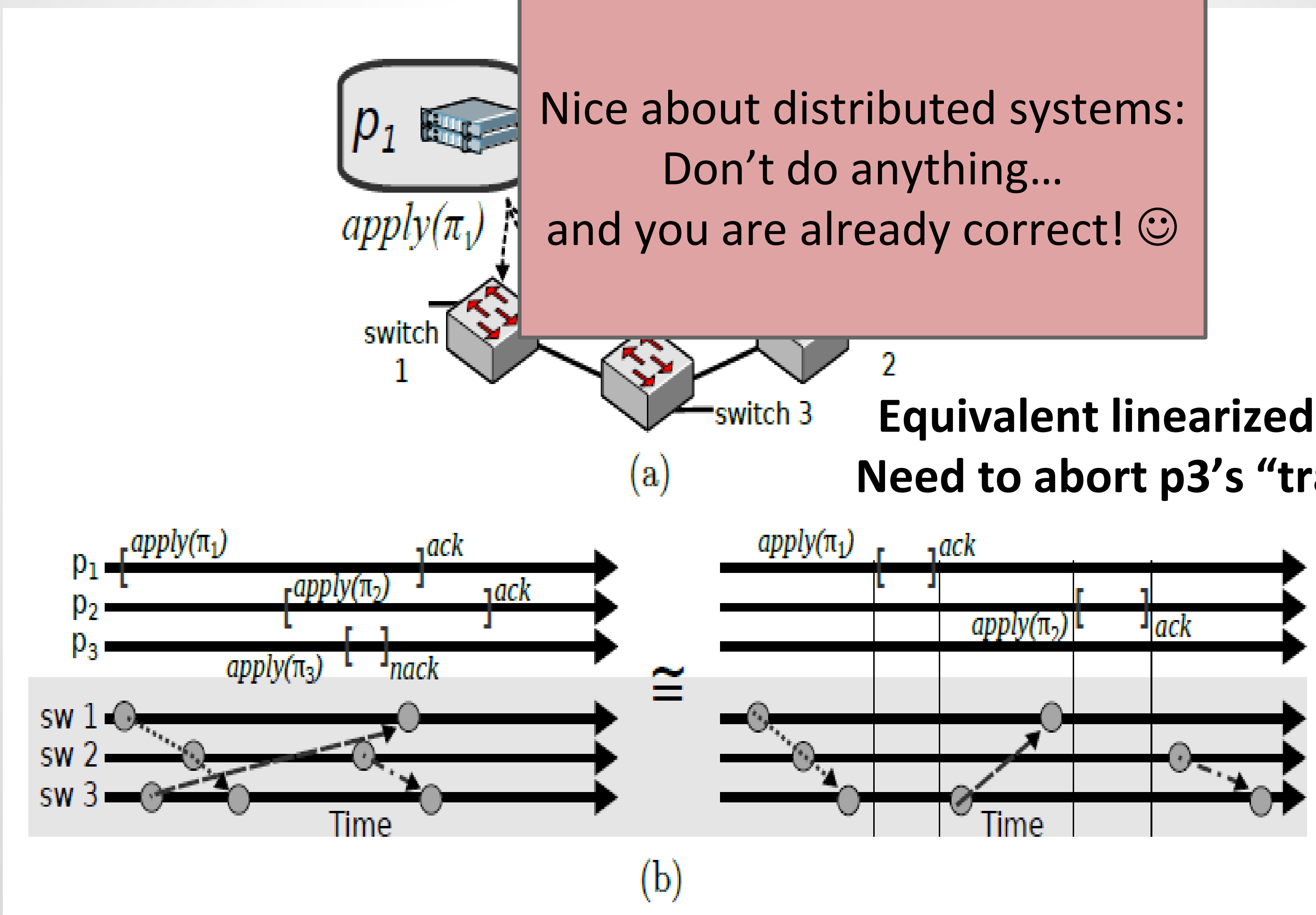
How to deal with concurrency?

A distributed systems problem!

Problem: Conflict free, per-packet consistent policy composition and installation

Holy Grails: Linearizability (Safety), Wait-freedom (Liveness)

Nice about distributed systems:
Don't do anything…
and you are already correct! ☺



Equivalent linearized schedule!
Need to abort p3's "transaction".

Reading / Literature Pointer

- 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.

- 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.

SDN: Dumb Switches?

How smart should switches be?

- Good news: separation of control plane
  - More global view
- Bad news: separation of control plane
  - Reduced visibility: in-band events?
  - What about the overhead / additional latency?

What is the right visibility?
Which functionality to keep in data plane?

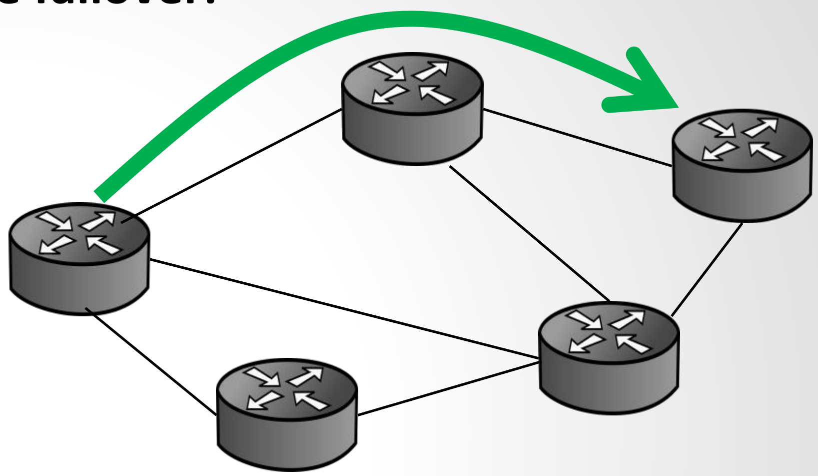
Example: where to implement robust routing?

Example: Fast Robust Routing Mechanisms

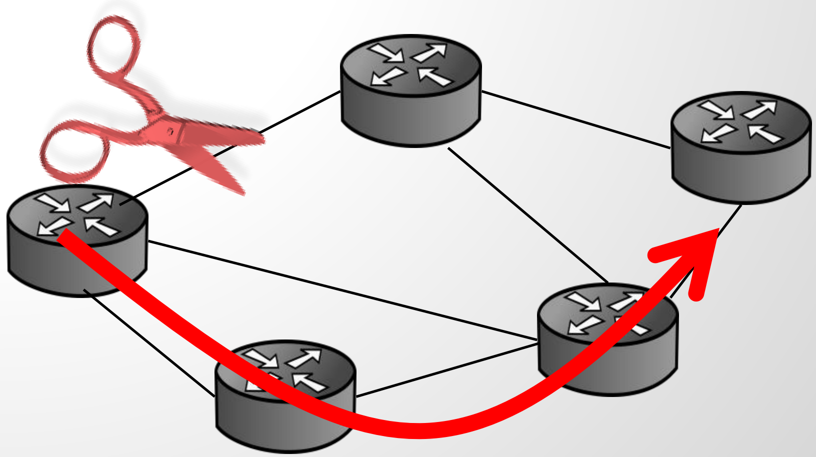
Modern networks provide **robust routing mechanisms**

- i.e., routing which reacts to failures
- example: MPLS local and global path protection

Before failover:



After failover:

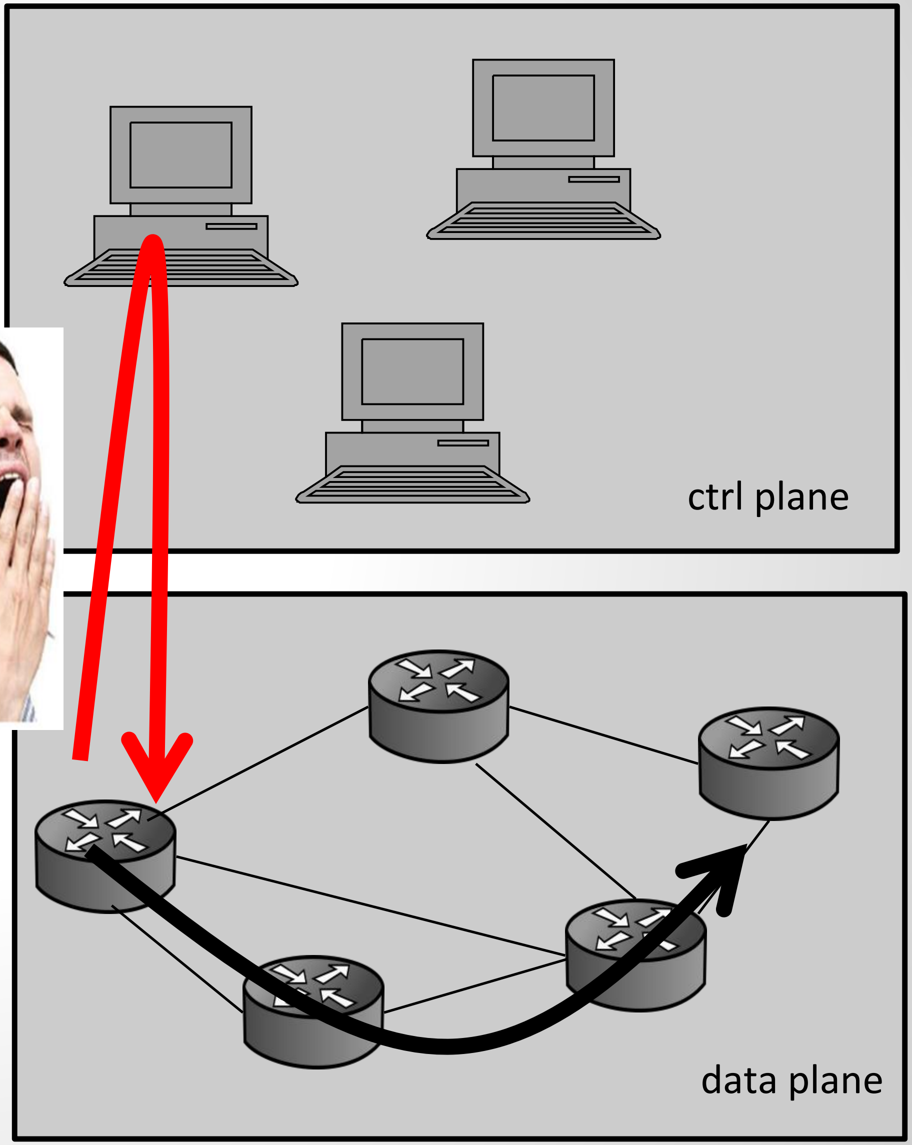


Fast In-band Failover

- Important that failover happens **fast = in-band**
  - Reaction time in control plane can be orders of magnitude slower

- For this reason: **OpenFlow Local Fast Failover Mechanism**
  - Supports conditional forwarding rules (depend on the local state of the link: live or not?)

- Gives fast but local and perhaps "**suboptimal**" forwarding sets
  - Controller improves globally later…



Reading / Literature Pointer

- Reclaiming the Brain: Useful OpenFlow Functions in the Data Plane
  Liron Schiff, Michael Borokhovich, and Stefan Schmid.
  13th ACM Workshop on Hot Topics in Networks (**HotNets**), Los Angeles, California, USA, October 2014.

- Provable Data Plane Connectivity with Local Fast Failover: Introducing OpenFlow Graph Algorithms
  Michael Borokhovich, Liron Schiff, and Stefan Schmid.
  ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (**HotSDN**), Chicago, Illinois, USA, August 2014.

What are use cases for SDN?
What are existing deployments?

SDN Use Cases

Many use cases discussed today, e.g. in:

- Enterprise networks
- Datacenters
- WANs
- IXPs
- ISPs

Existing deployments!

The origins? E.g., Stanford campus network (coined SDN)

Many use cases discussed today, e.g. in:

- Enterprise networks
- Datacenters
- WANs
- IXPs
- ISPs

Existing deployments!

The killer application? Network virtualization

Deployments at Microsoft and Google

The origins? E.g., Stanford campus network (coined SDN)

Many use cases discussed today, e.g. in:

- Enterprise networks
- Datacenters
- WANs
- IXPs
- ISPs

Existing deployments!

The killer application? Network virtualization

Deployments at Microsoft and Google

What is it used for?
And how is it deployed?

The origins? E.g., Stanford campus network (coined SDN)

3

Many use cases discussed today, e.g. in:

- Enterprise networks
- Datacenters
- WANs
- IXPs
- ISPs

Existing deployments!

The killer application? Network virtualization

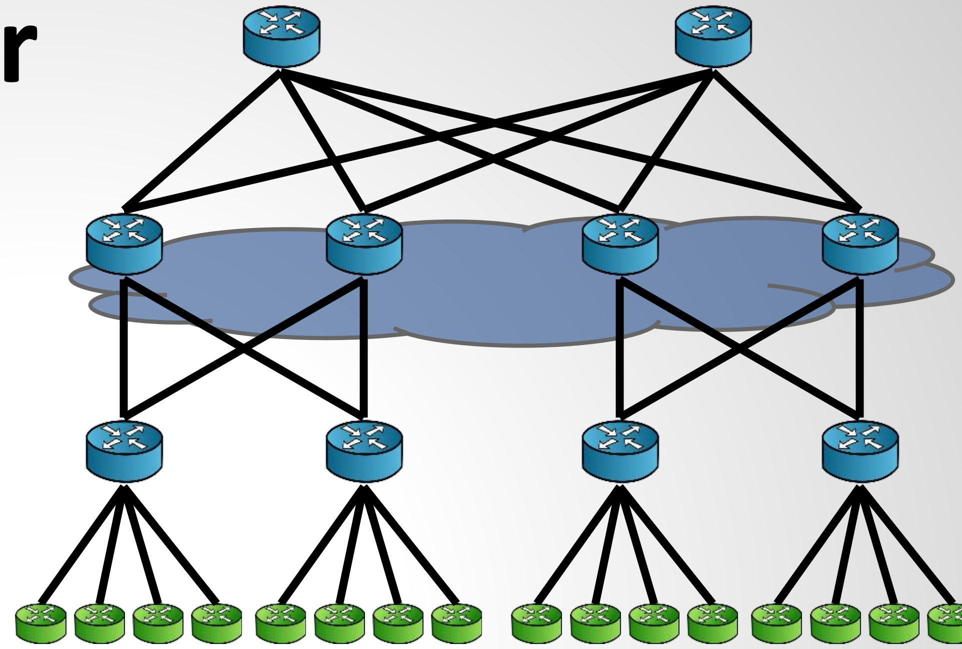
1

Deployments at Microsoft and Google

2

What is it used for?
And how is it deployed?

SDN in Datacenter



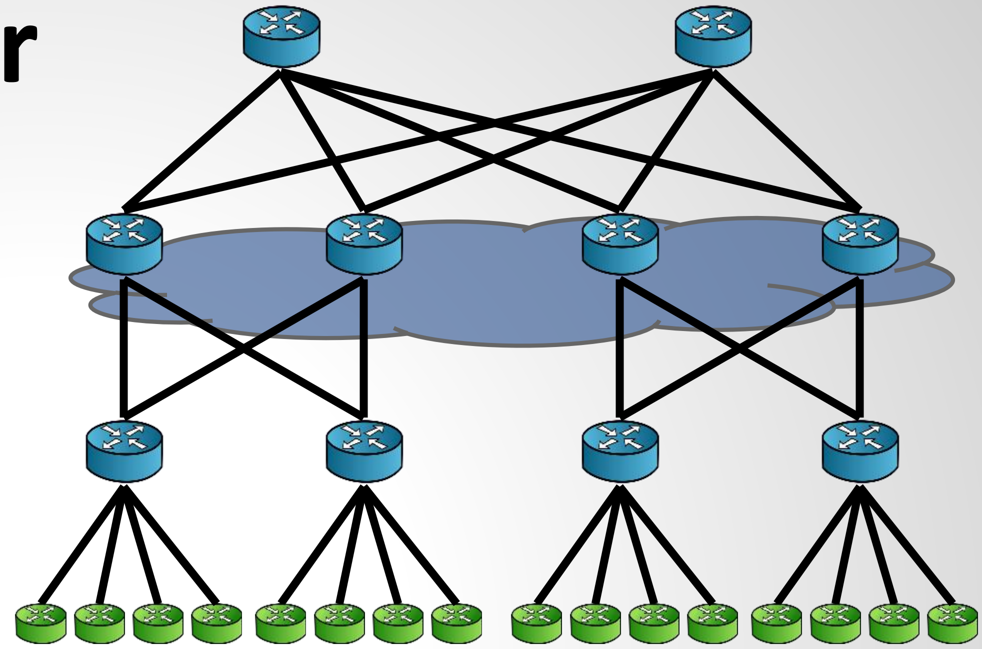
Characteristics

- Already highly virtualized
- Quite homogeneous
- Scalability a challenge

Why SDN?

How?

SDN in Datacenter



Characteristics

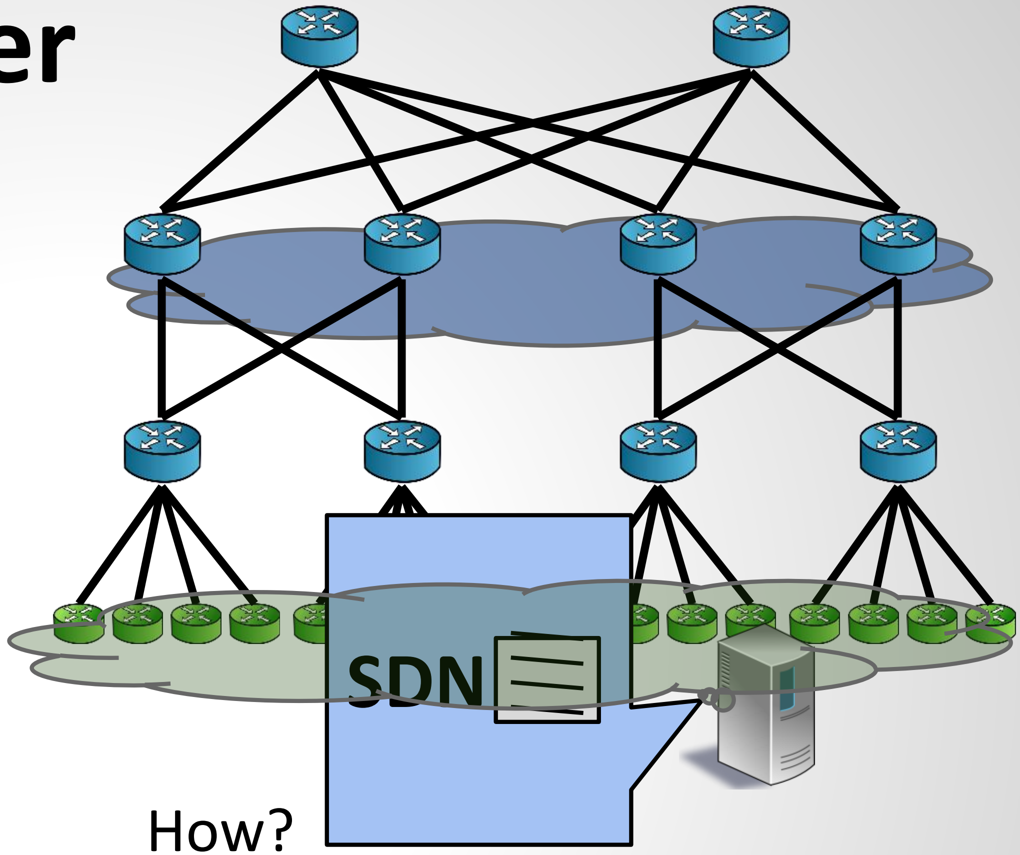
- Already highly virtualized
- Quite homogeneous
- Scalability a challenge

Why SDN?

- Decouple application from physical infrastructure
- Enable virtual networks (e.g., Nicira): own addresses for tenants, isolation, support for seamless VM migration
- Performance: improve throughput

How?

SDN in Datacenter



Characteristics

- Already highly virtualized
- Quite homogeneous
- Scalability a challenge

Why SDN?

- Decouple application from physical infrastructure
- Enable virtual networks (e.g., Nicira): own addresses for tenants, isolation, support for seamless VM migration
- Performance: improve throughput

Excursion: Scalability Challenge (1)

- A single datacenter can consist of
  - 100k servers…
  - … à 32 VMs each
  - So 3.2 million VMs!
- Each VM requires a MAC and IP address
- But also physical infrastructure needs addresses

Slide © James Kempf, Ericsson Research

Excursion: Scalability Challenge (2)

- Broadcast domains needed
  - ARP and Neighbor Discovery (ND) for resolving IP to MAC address can be overwhelming
  - Broadcast, multicast, …
- In addition: require
  - Isolation between different tenants (addresses, traffic)
  - Support for dynamic provisioning and migration (e.g., maintenance, optimization)

Slide © James Kempf, Ericsson Research

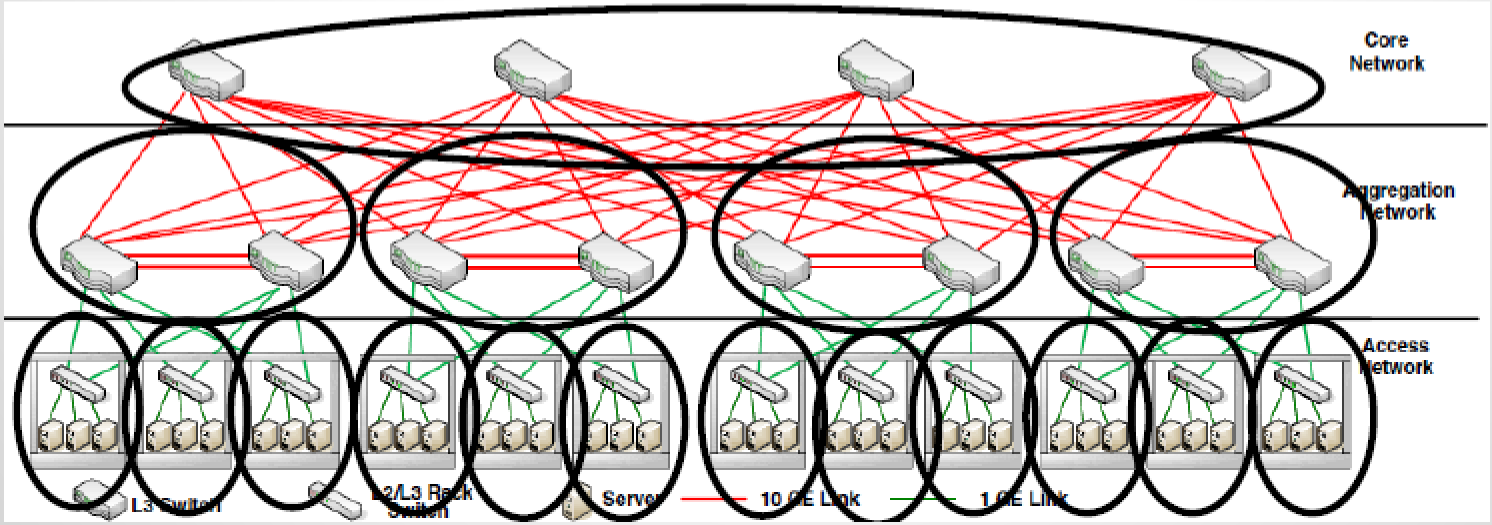
Excursion: Virtualization

- Classically 2 solutions to provide isolation
- VLANs, but:
  - Only 4096 tags
  - For dynamic VM placement, each server-TOR link must be configured as a trunk
  - Can move VMs only to servers where VLAN tag is configured
  - Physical dependencies, dynamic trunking difficult…
  - Often used in single-tenant datacenters (e.g., Ericsson)
- Overlays better but:
  - E.g., VPLS or VPNs
  - But challenging to coordinate overlay and underlay

Slide © James Kempf, Ericsson Research

Excursion: Virtualization

- Classic approaches: IP subnets, VLANs, Overlays
- IP Subnets: last hop router?
  - If at TOR: broadcast limited to rack, poor mobility
  - If at Core: high mobility but unlimited broadcast



Slide © James Kempf, Ericsson Research

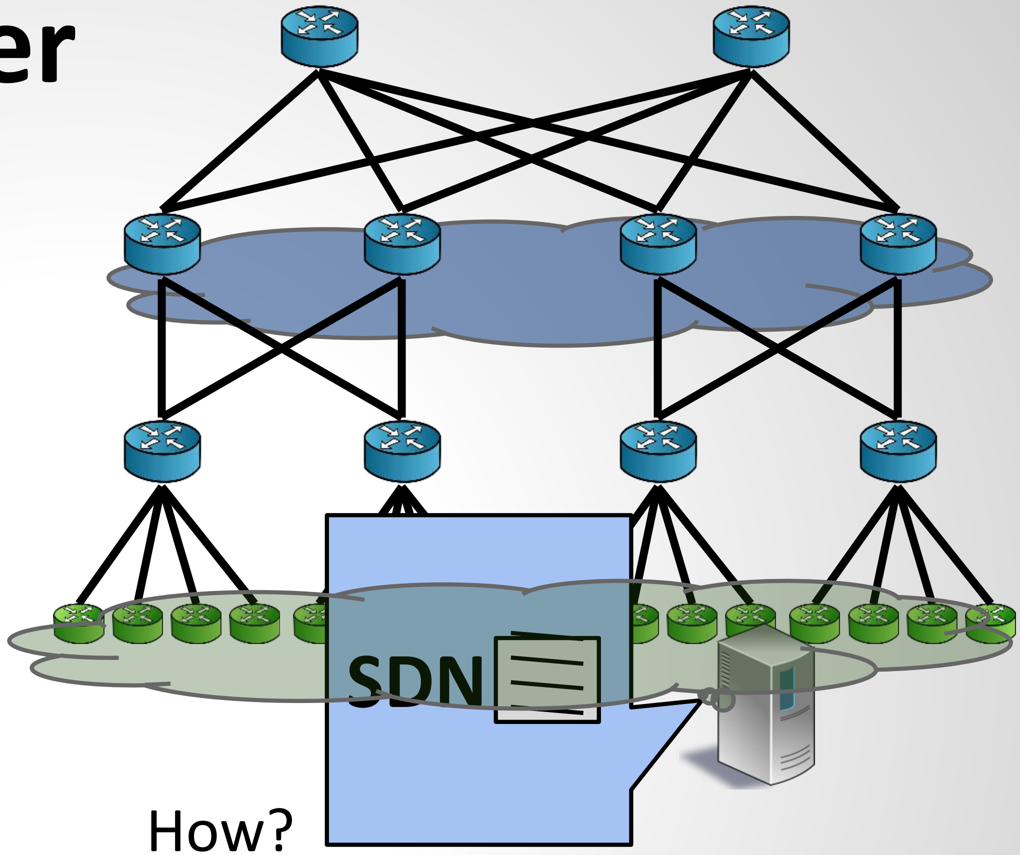
SDN in Datacenter

Characteristics

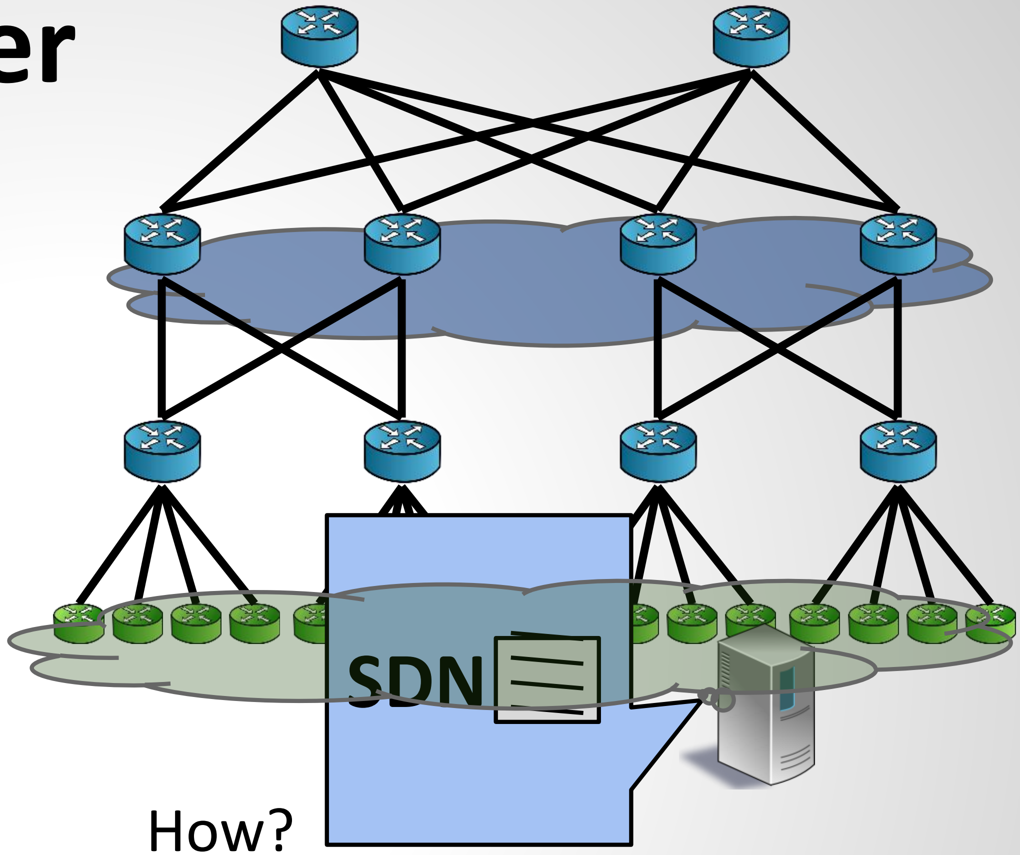
- Already highly virtualized
- Quite homogeneous
- Scalability a challenge

Why SDN?

- Decouple application from physical infrastructure
- Enable virtual networks (e.g., Nicira): own addresses for tenants, isolation, support for seamless VM migration
- Performance: improve throughput



SDN in Datacenter



Characteristics

- Already highly virtualized
- Quite homogeneous
- Scalability a challenge

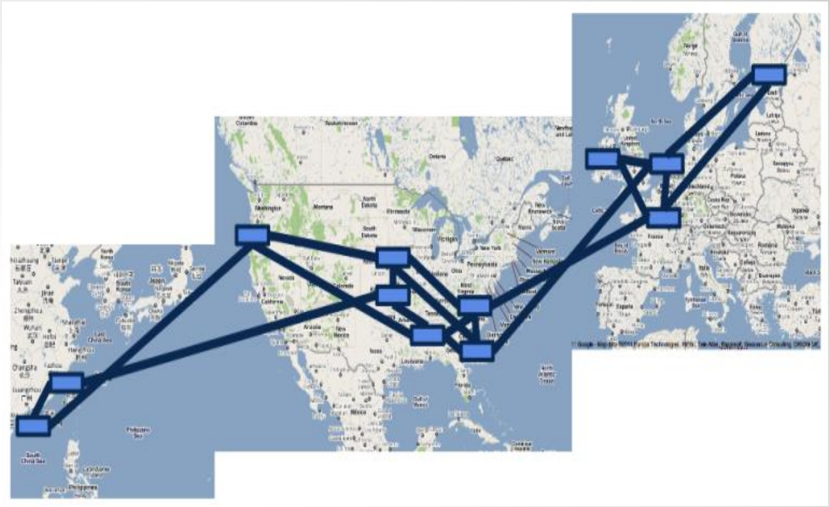
Why SDN?

- Decouple application from physical infrastructure
- Enable virtual networks (e.g., Nicira): own addresses for tenants, isolation, support for seamless VM migration
- Performance: improve throughput

How?

- Two separate control planes at edge and in core; «innovation» only at edge
- Provide simple fabric abstraction to tenant: classify packets at ingress and providing tunnels (through ECMP fabric)
- SDN deployment easy: software switches (Open vSwitch) at edge, software update

SDN in WAN



Characteristics

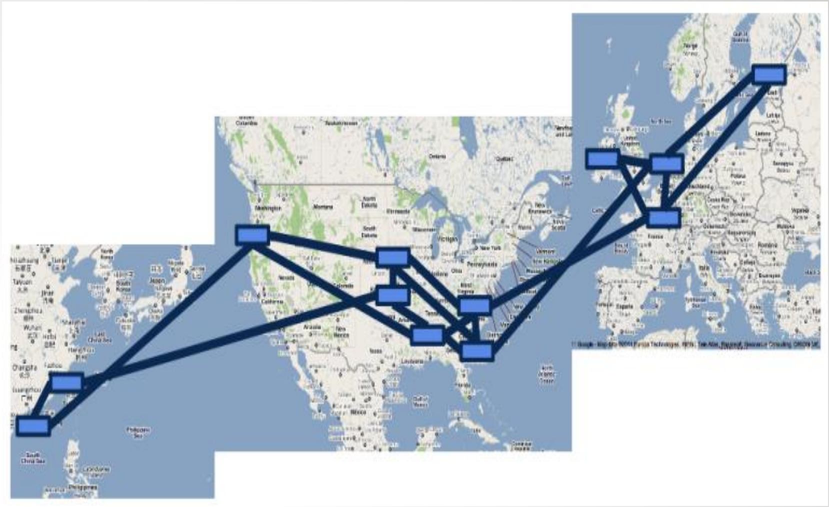
Why SDN?

How?

SDN in WAN

Characteristics

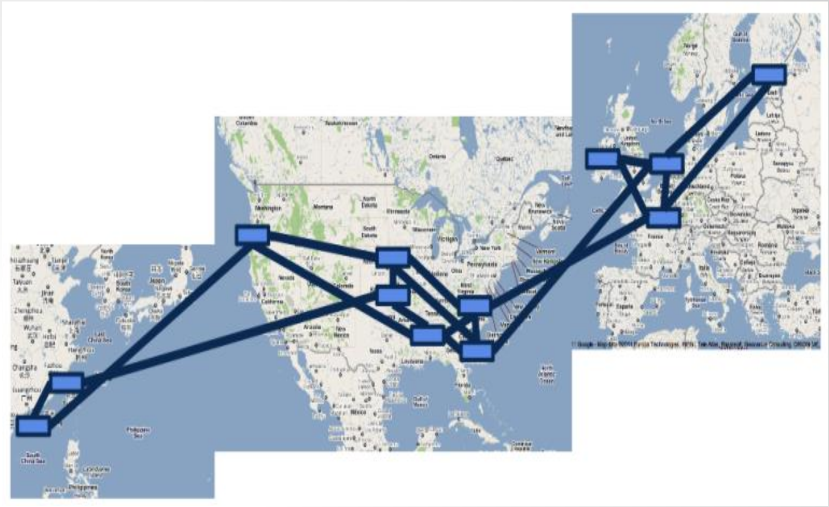
- Small: not many sites
- Many different applications and requirements, latency matters
- Bandwidth precious (WAN traffic grows fastest): 1G fiber connection from San Jose costs USD 3000/month



Why SDN?

How?

SDN in WAN



Characteristics

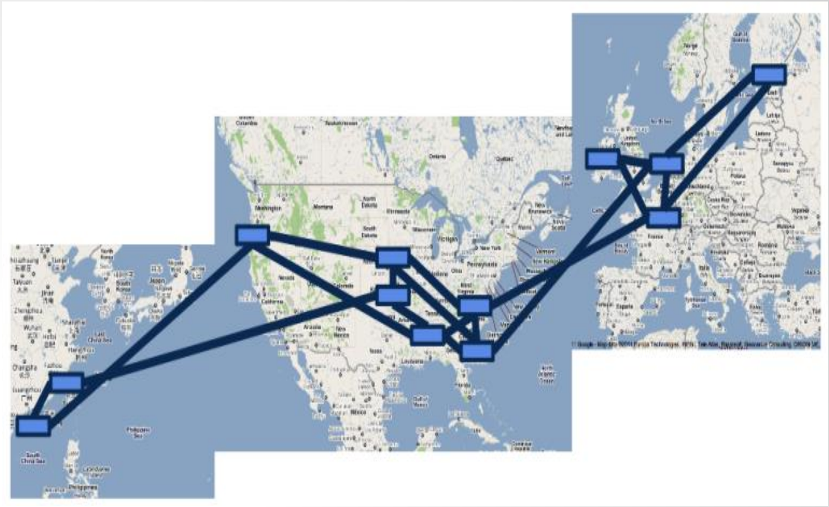
- Small: not many sites
- Many different applications and requirements, latency matters
- Bandwidth precious (WAN traffic grows fastest): 1G fiber connection from San Jose costs USD 3000/month

Why SDN?

- Improve utilization (e.g., Google B4) and safe costs (e.g., Microsoft SWAN)
- Differentiate applications (latency sensitive Google docs vs datacenter synchronization)

How?

SDN in WAN



Characteristics

- Small: not many sites
- Many different applications and requirements, latency matters
- Bandwidth precious (WAN traffic grows fastest): 1G fiber connection from San Jose costs USD 3000/month

Why SDN?

- Improve utilization (e.g., Google B4) and safe costs (e.g., Microsoft SWAN)
- Differentiate applications (latency sensitive Google docs vs datacenter synchronization)

How?

- Replace IP “core” routers (running BGP) at border of datacenter (end of long-haul fiber)
- Gradually replace routers

SDN in WAN

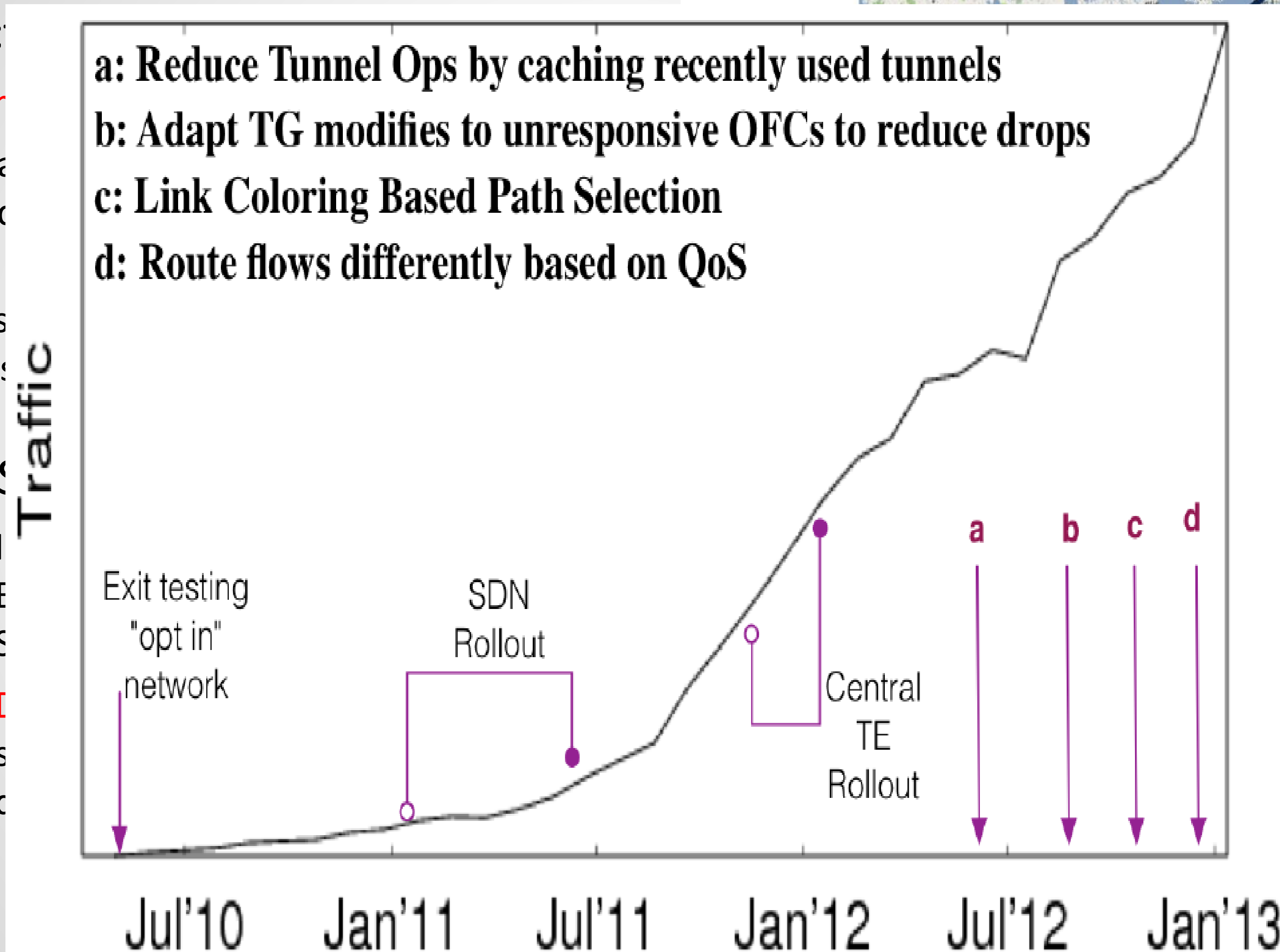
Charac

- Sm
- Ma
  rec
- Ba
  fas
  cos

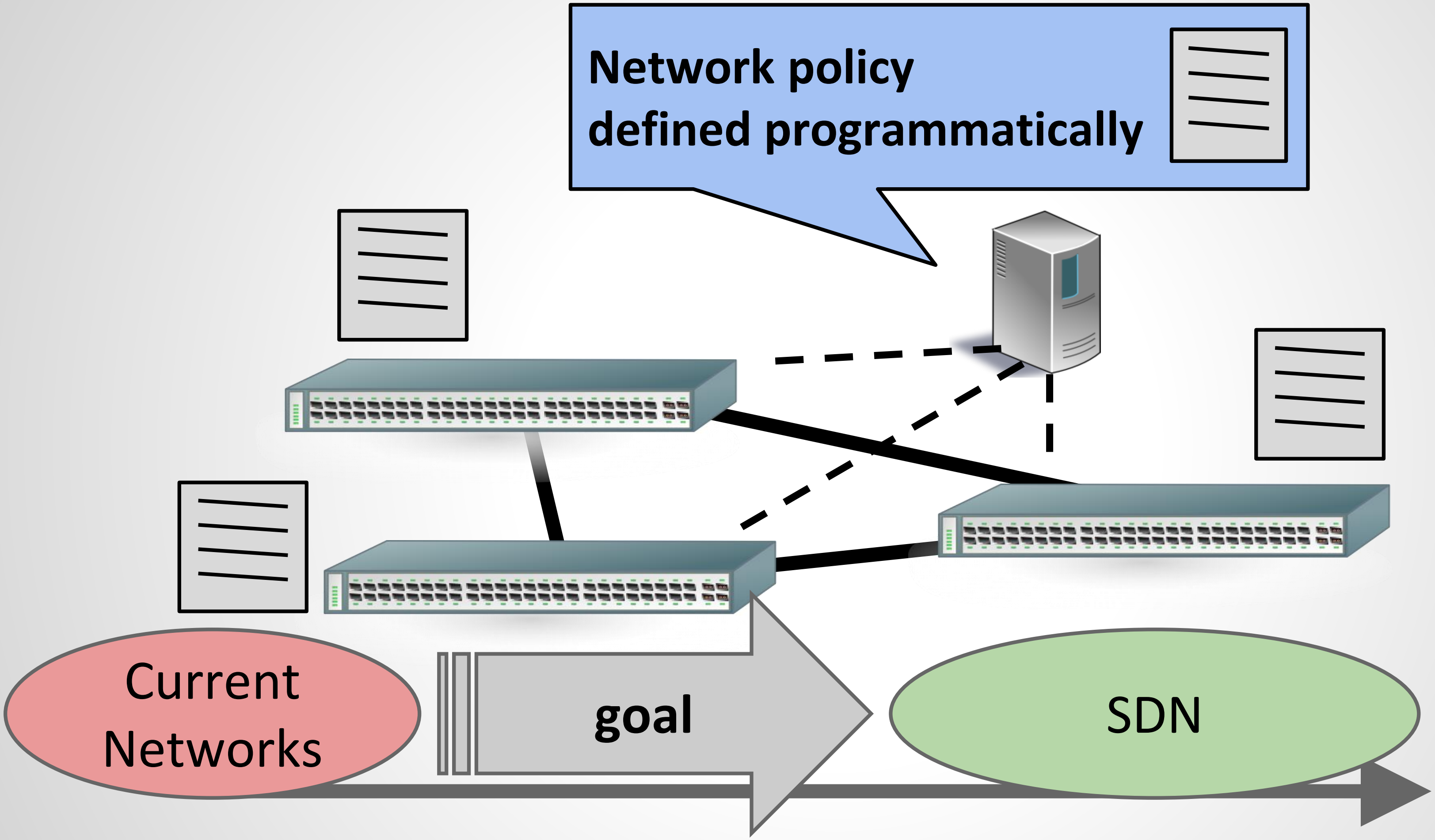
Why S

- I
  E
  S
- D
  s
  c

ng
nd



Use Case: Why SDN in Enterprise?

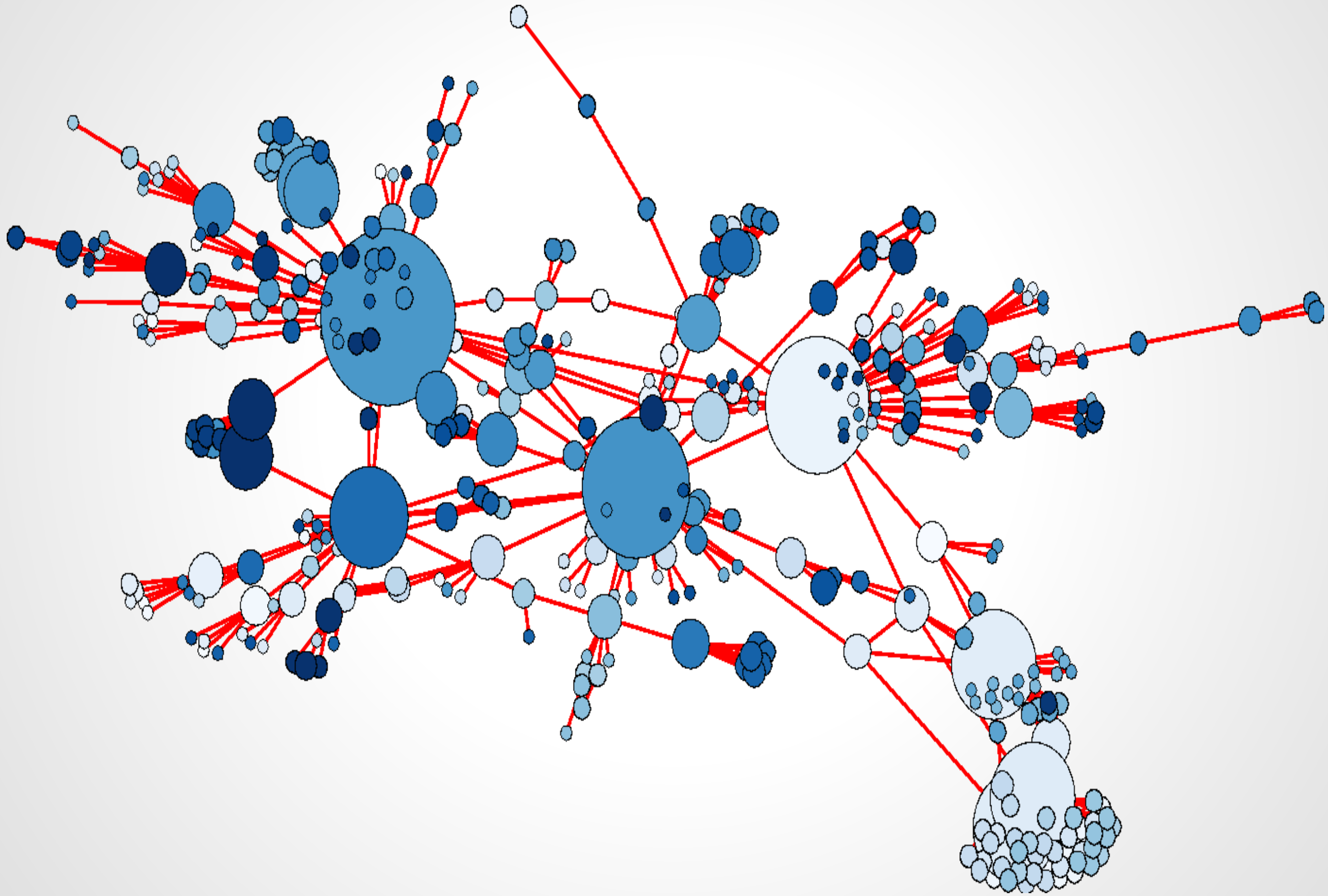


Main benefit: automation and abstraction for networks

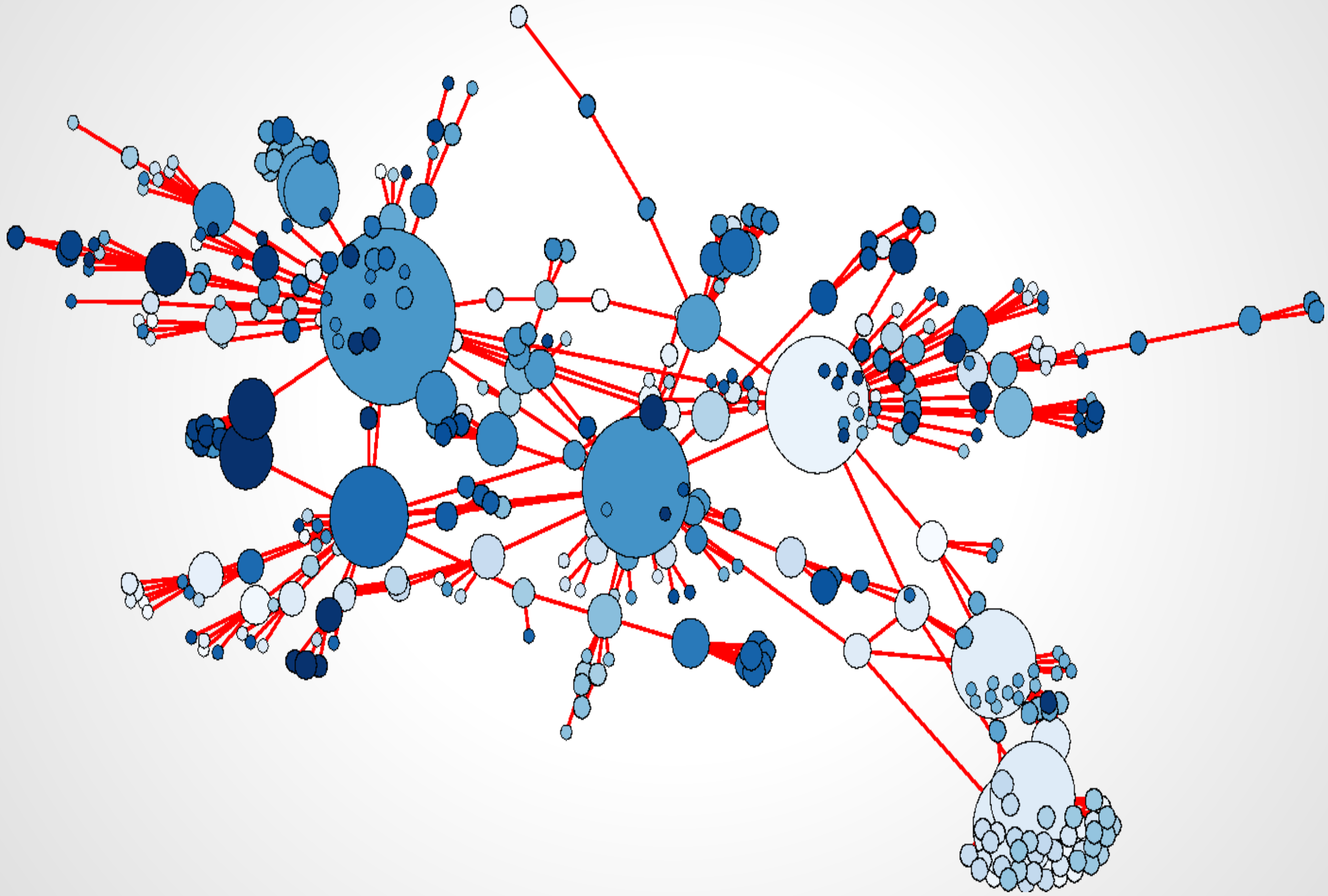
But how to deploy SDN in enterprise?

- Infrastructure **budgets are limited**
- Idea: Can we **incrementally deploy SDN** into enterprise campus networks?
- And what **SDN benefits** can be realized in a hybrid deployment?

Can we deploy SDN in enterprise edge?

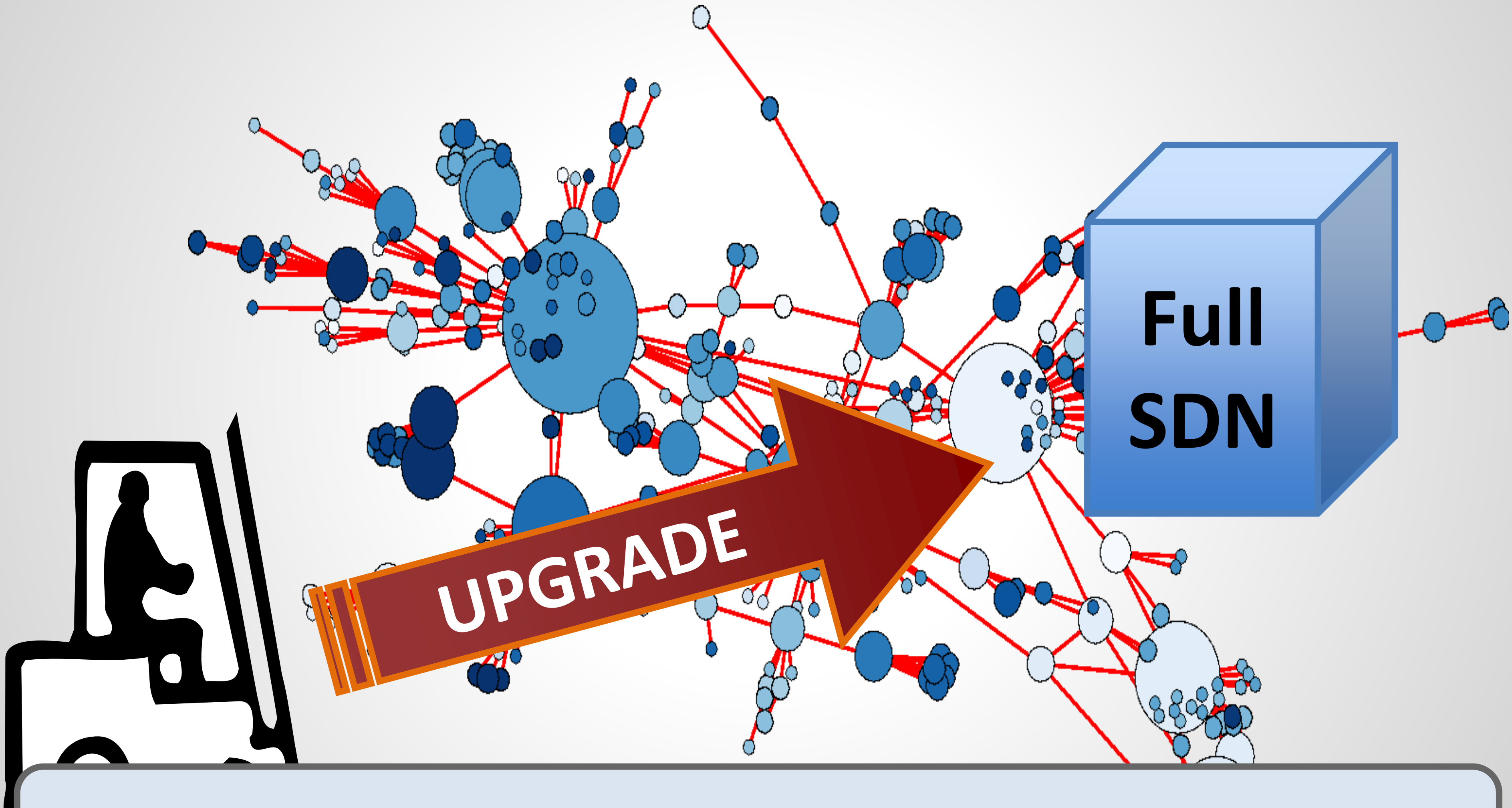


Can we deploy SDN in enterprise edge?



The edge is large, and not in software!

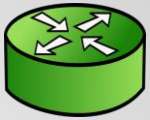
The SDN Deployment Problem

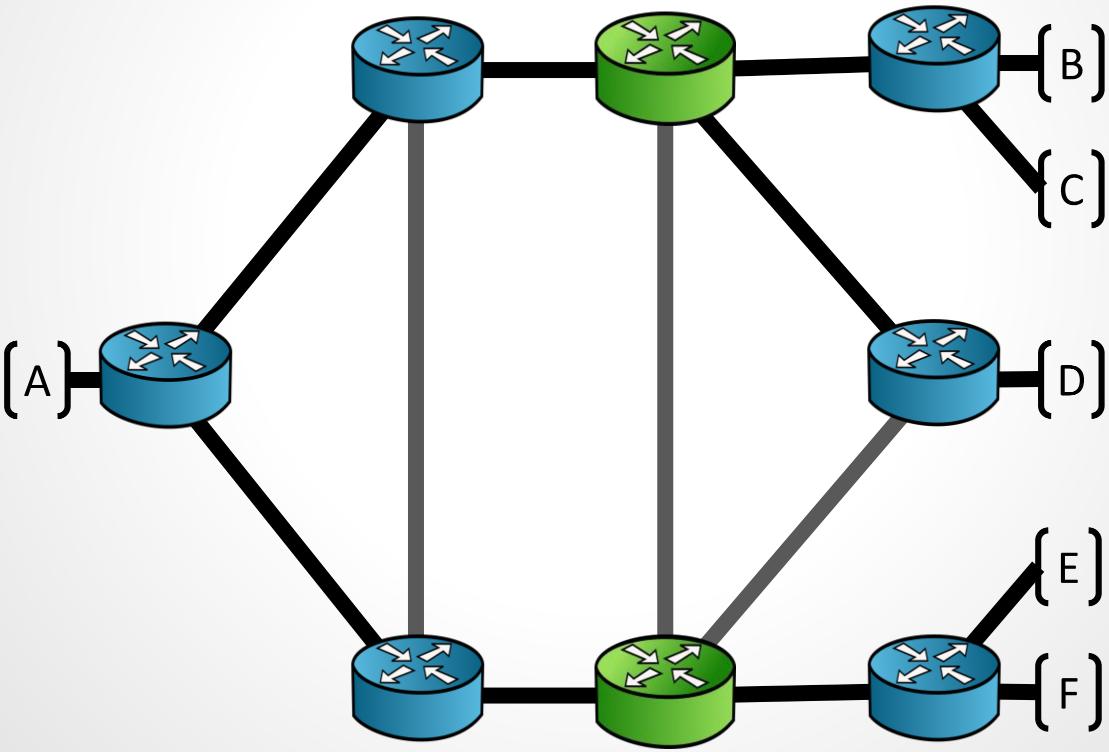


Expensive and undesired: must upgrade to SDN incrementally

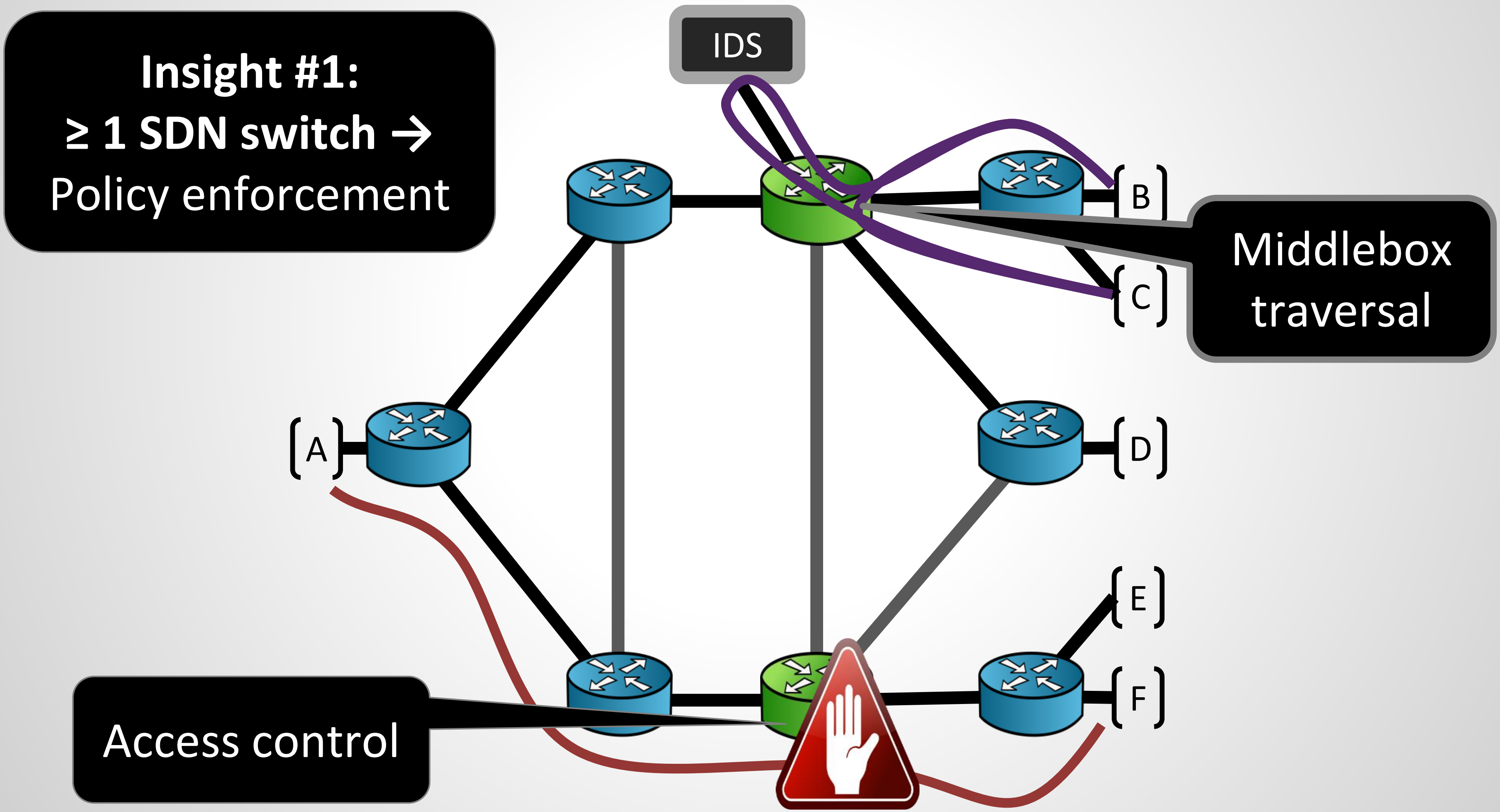
Key Questions

- How can we **incrementally deploy SDN** into enterprise campus networks?
- What **SDN benefits** can be realized in a hybrid deployment?

The Partial SDN Deployment ()

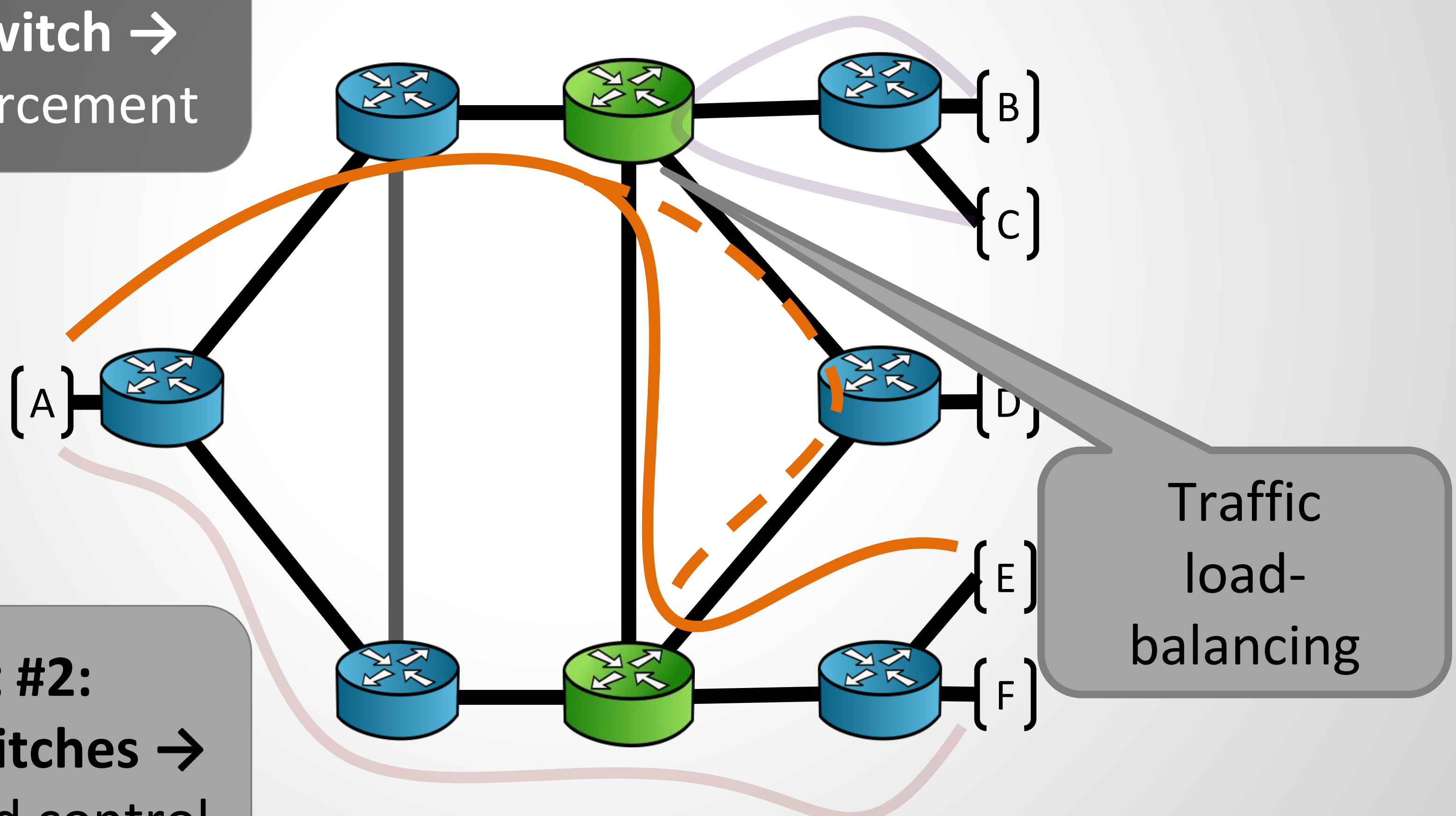


Get Functionality with Waypoint Enforcement



Larger Deployment = More Flexibility

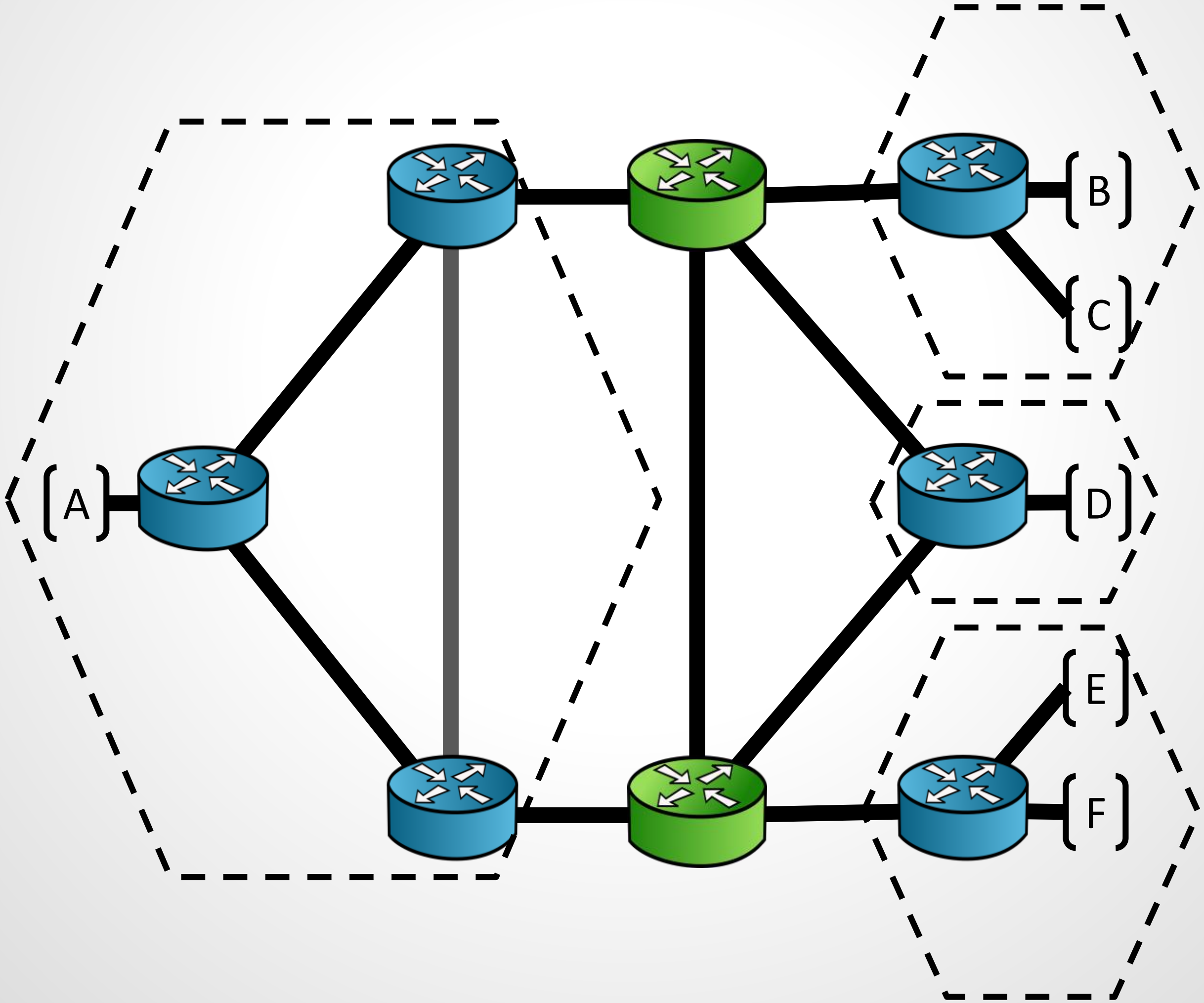
Insight #1:
≥ 1 SDN switch →
Policy enforcement



Insight #2:
≥ 2 SDN switches →
Fine-grained control

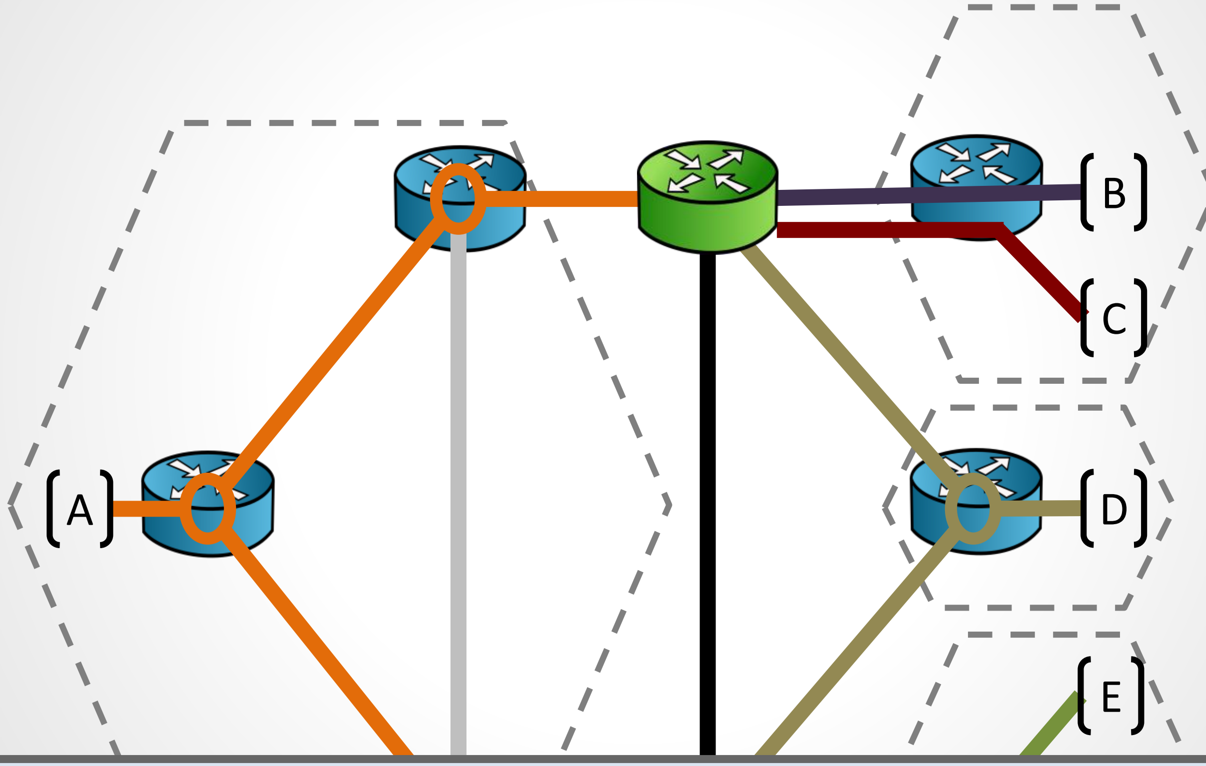
Panopticon: Building the Logical SDN Abstraction

1. Group SDN ports in **Cell Blocks**



Panopticon: Building the Logical SDN Abstraction

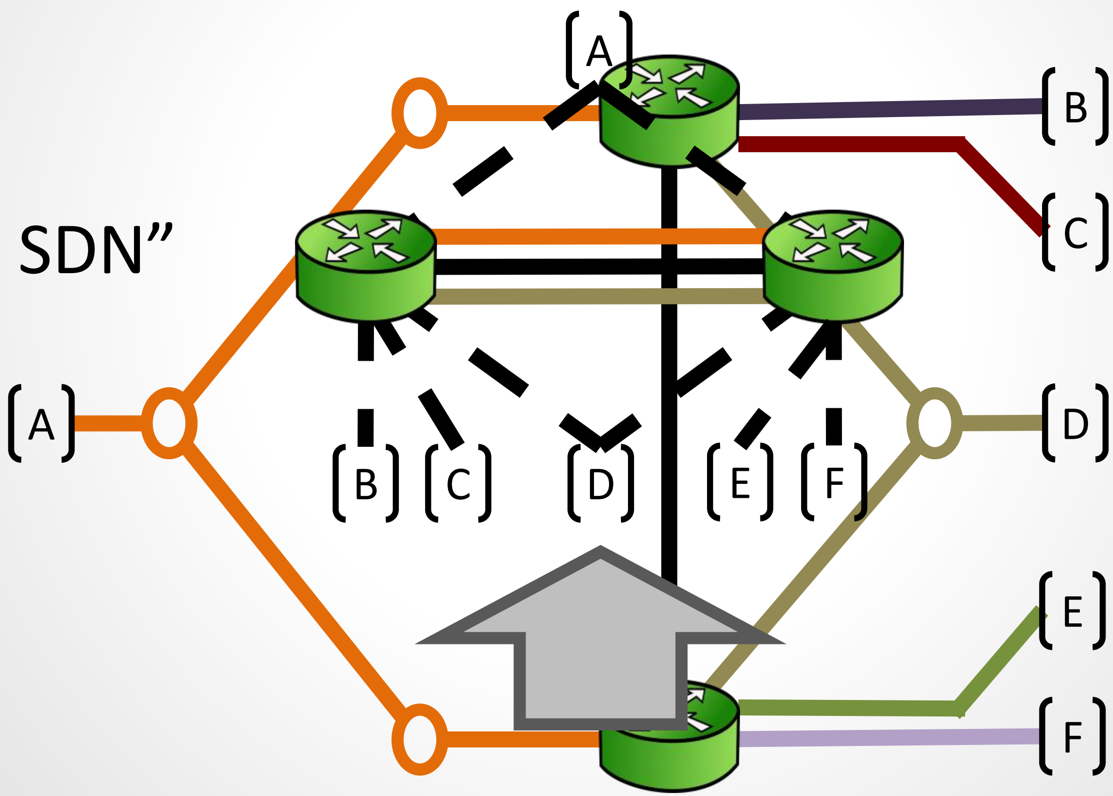
2. Restrict traffic by using VLANs

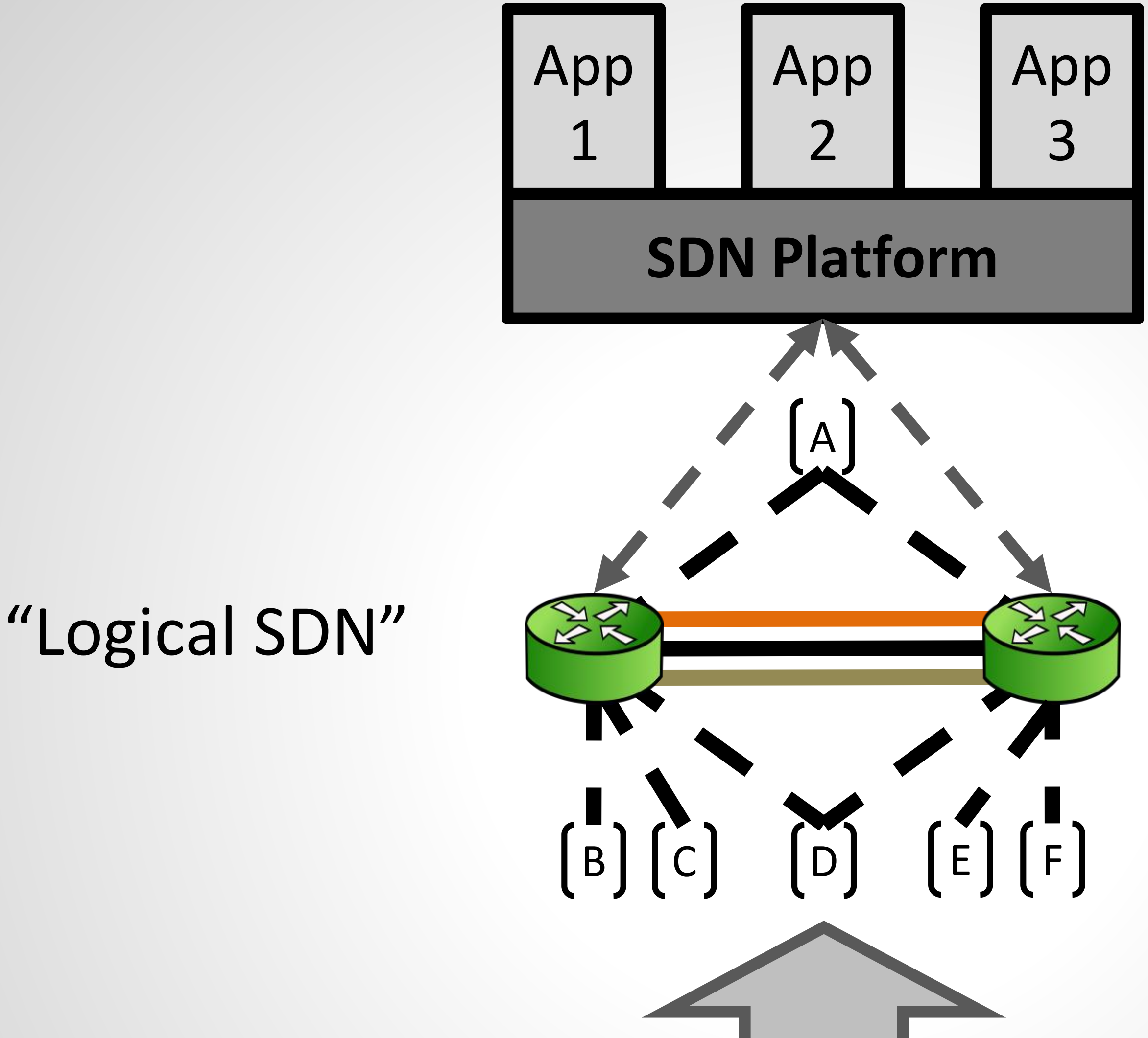


Per-port spanning trees that ensure waypoint enforcement

Panopticon: Building the Logical SDN Abstraction

"Logical SDN"



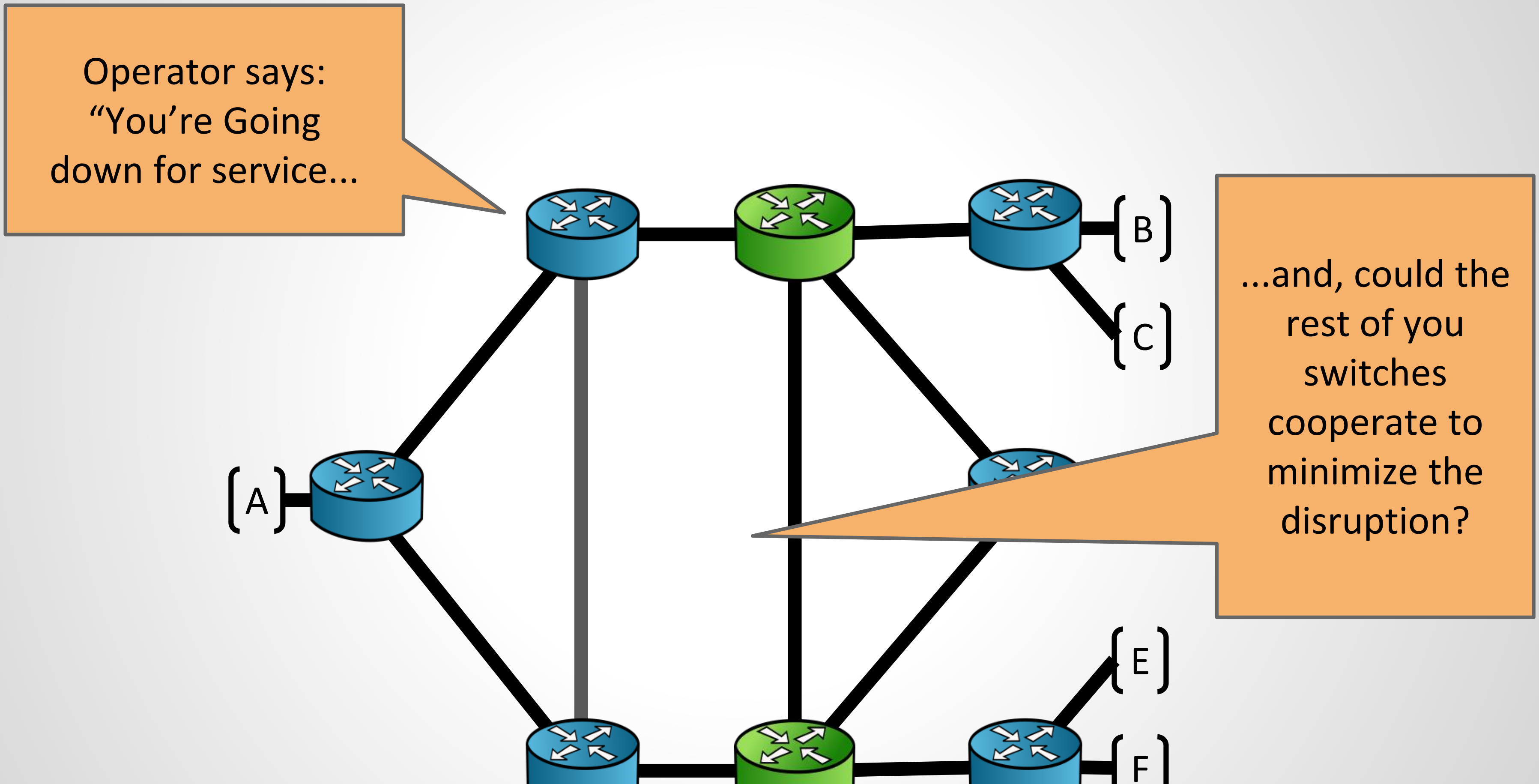


PANOPTICON provides the abstraction of a (nearly) fully-deployed SDN in a partially upgraded network

What is the value of
a logical SDN

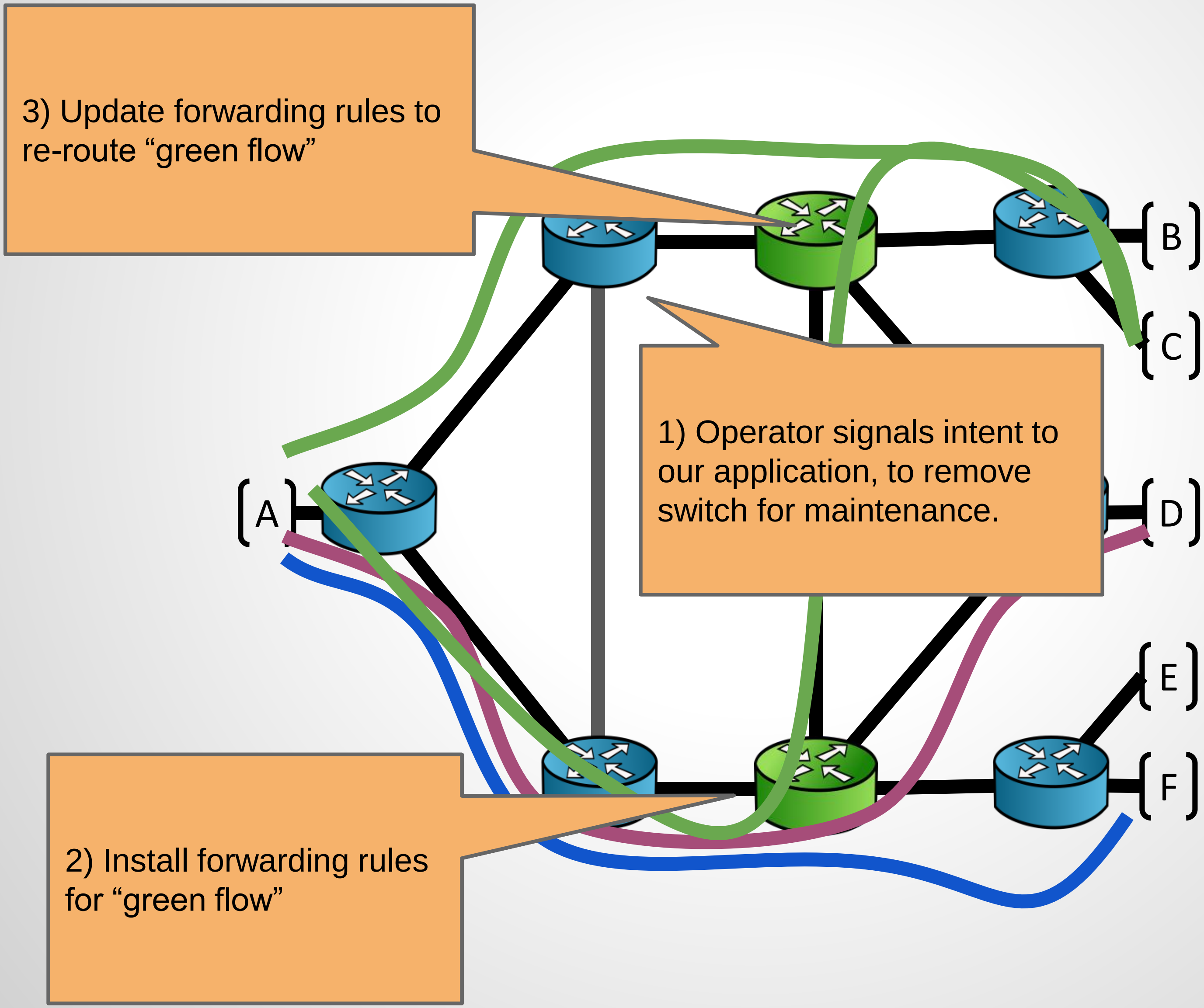
?

Use Case 1: Planned Maintenance



Let software worry about the dependencies, not the human operator!

Use Case 1: Planned Maintenance



Reading / Literature Pointer

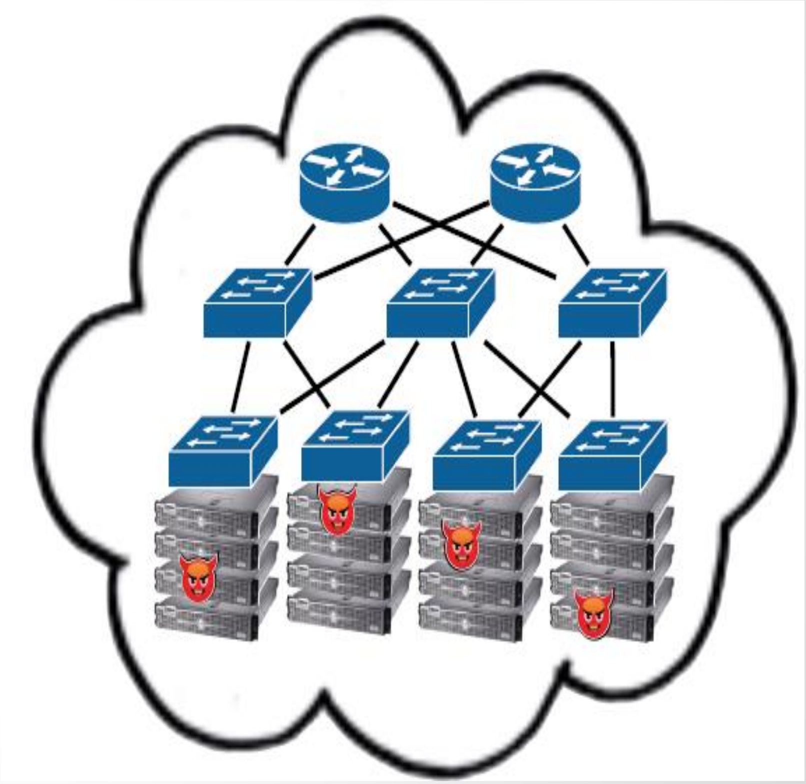
- Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks
  Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, and Anja Feldmann.
  USENIX Annual Technical Conference (**ATC**), Philadelphia, Pennsylvania, USA, June 2014.

SDN: Easy to make network adaptive?

Not really: consistency is a challenge!

Example: Cloud

What if your traffic was *not* isolated from other tenants during periods of routine maintenance?



Example: Outages

Even technically sophisticated companies are struggling to build networks that provide reliable performance.

github SOCIAL CODING

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

amazon web services

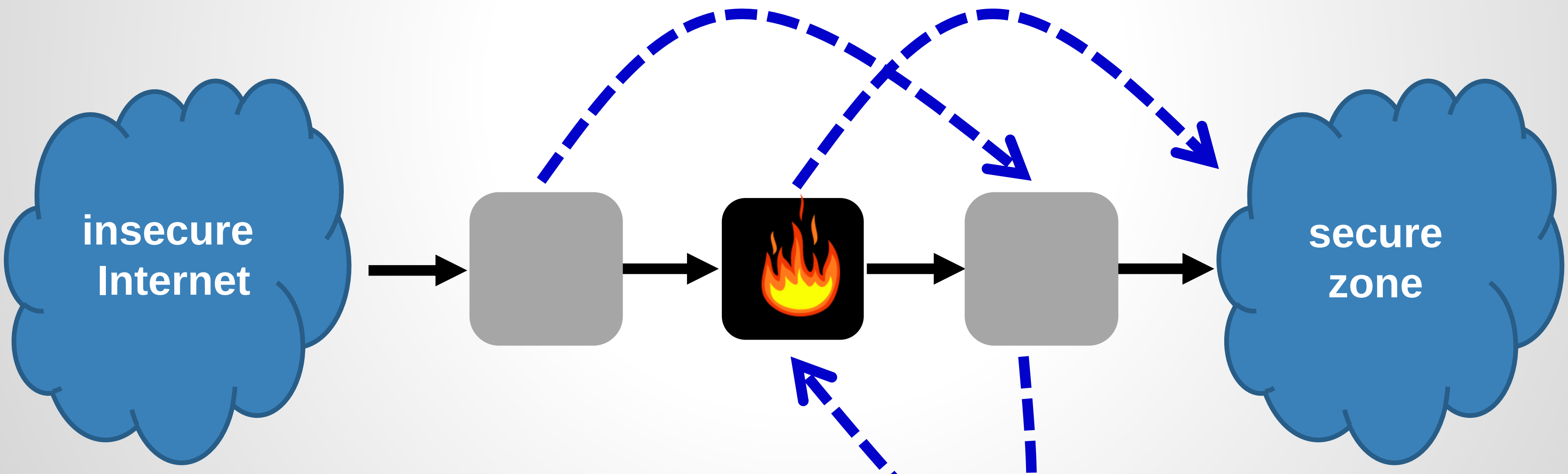
GoDaddy.com

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

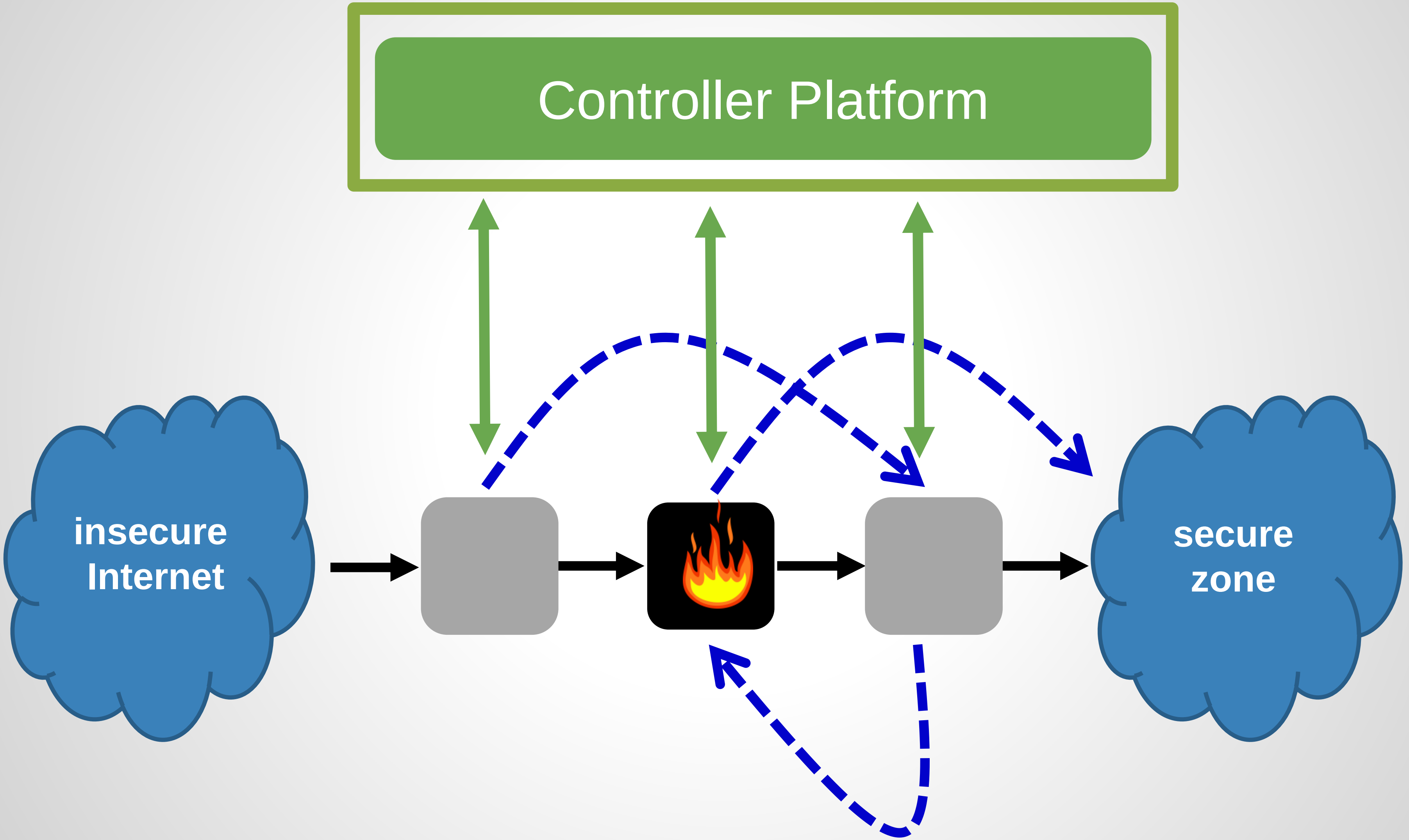
Experienced a network connectivity issue [...] interrupted the airline's flight departures, airport processing and reservations systems

United Airlines

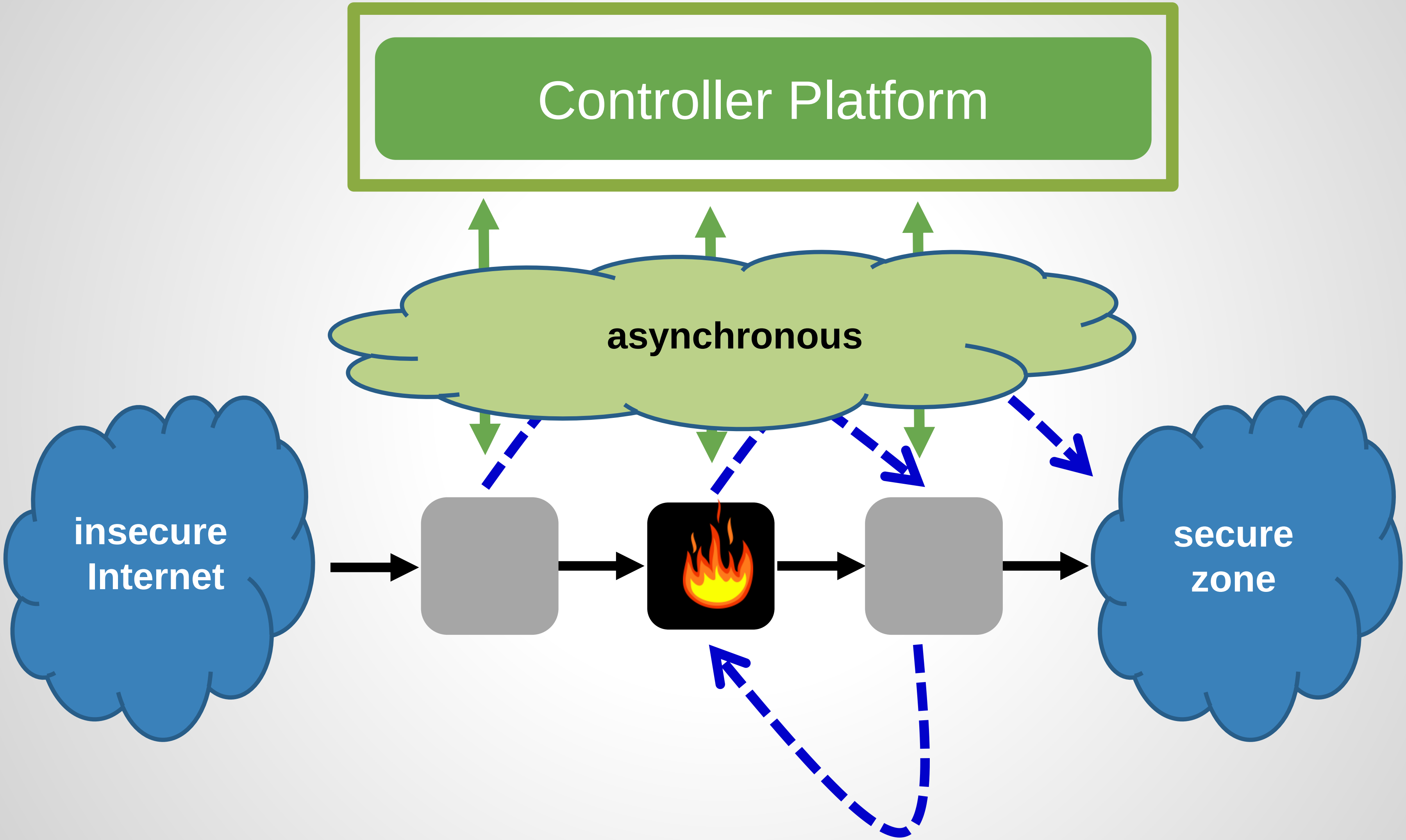
Example: Security-Critical Updates



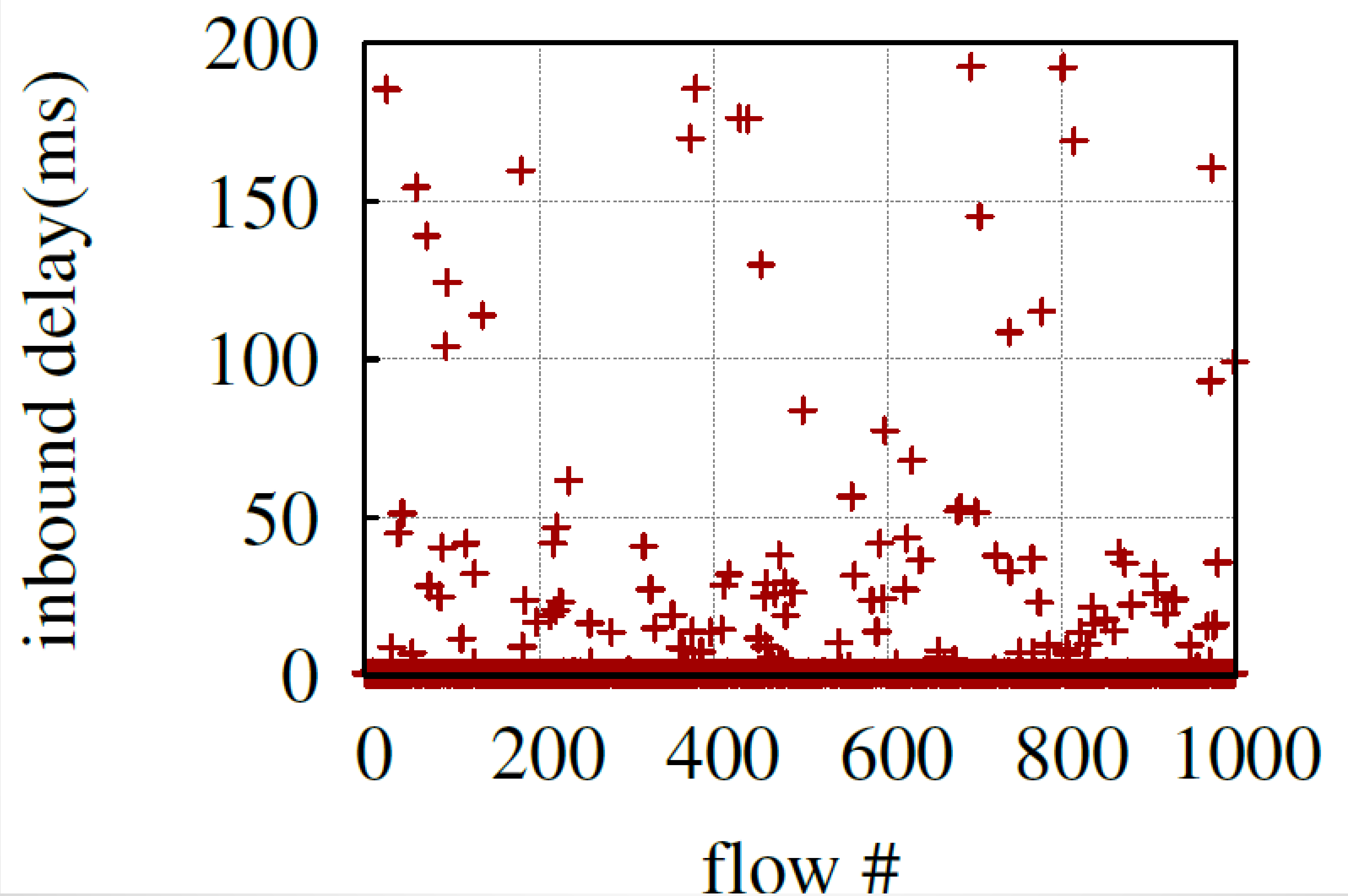
Example: Security-Critical Updates



Example: Security-Critical Updates



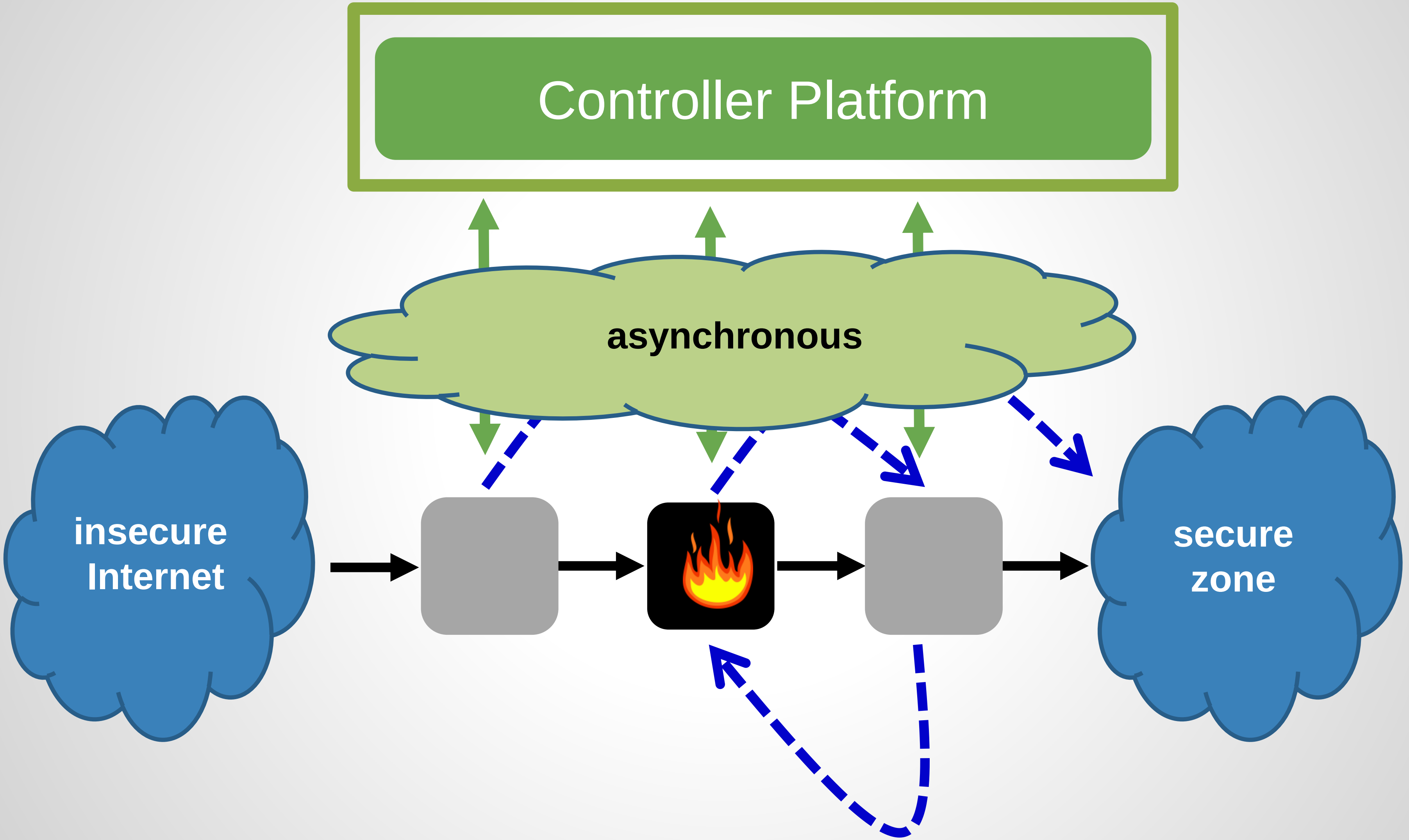
An Asynchronous Distributed System



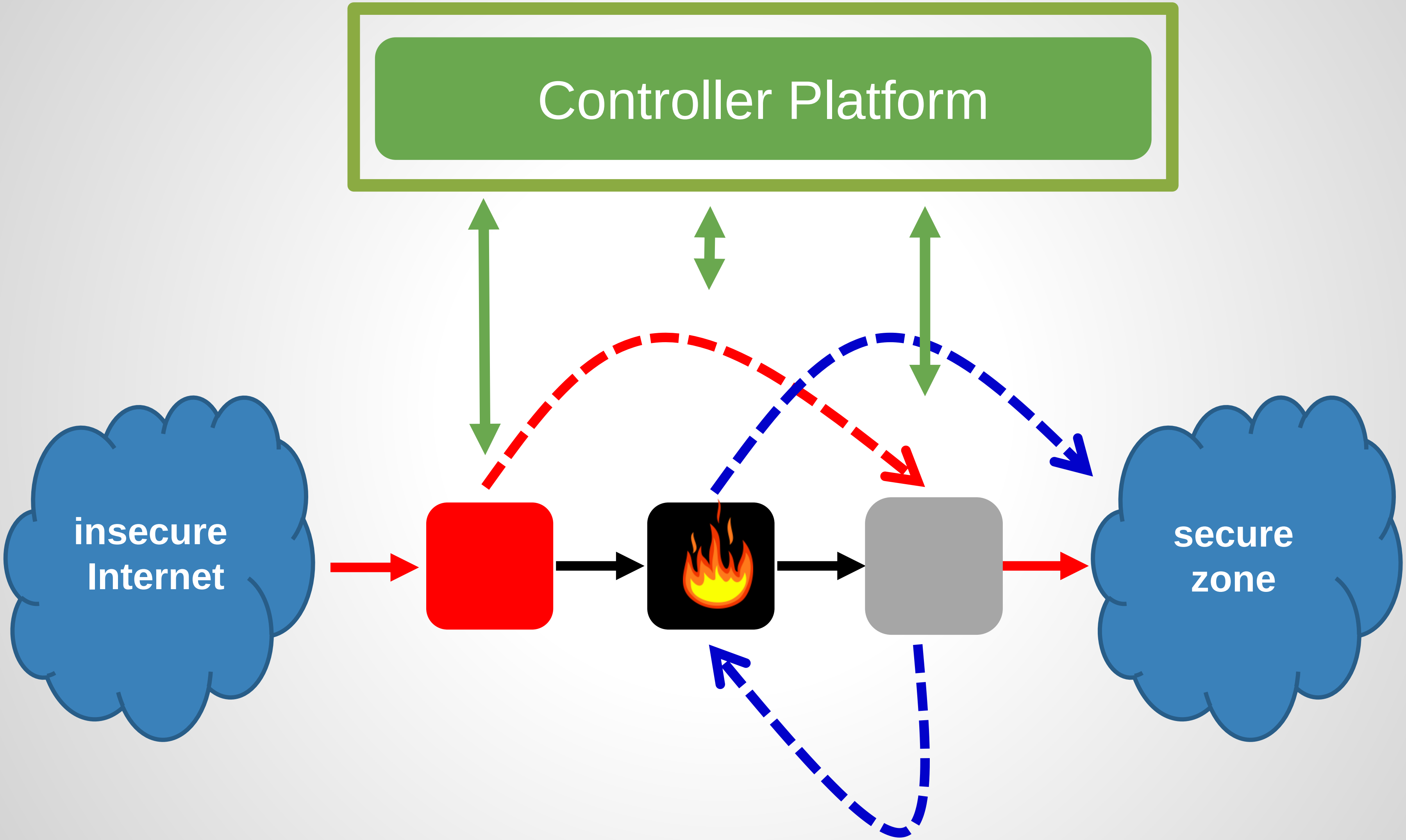
He et al., ACM SOSR 2015:

without network latency

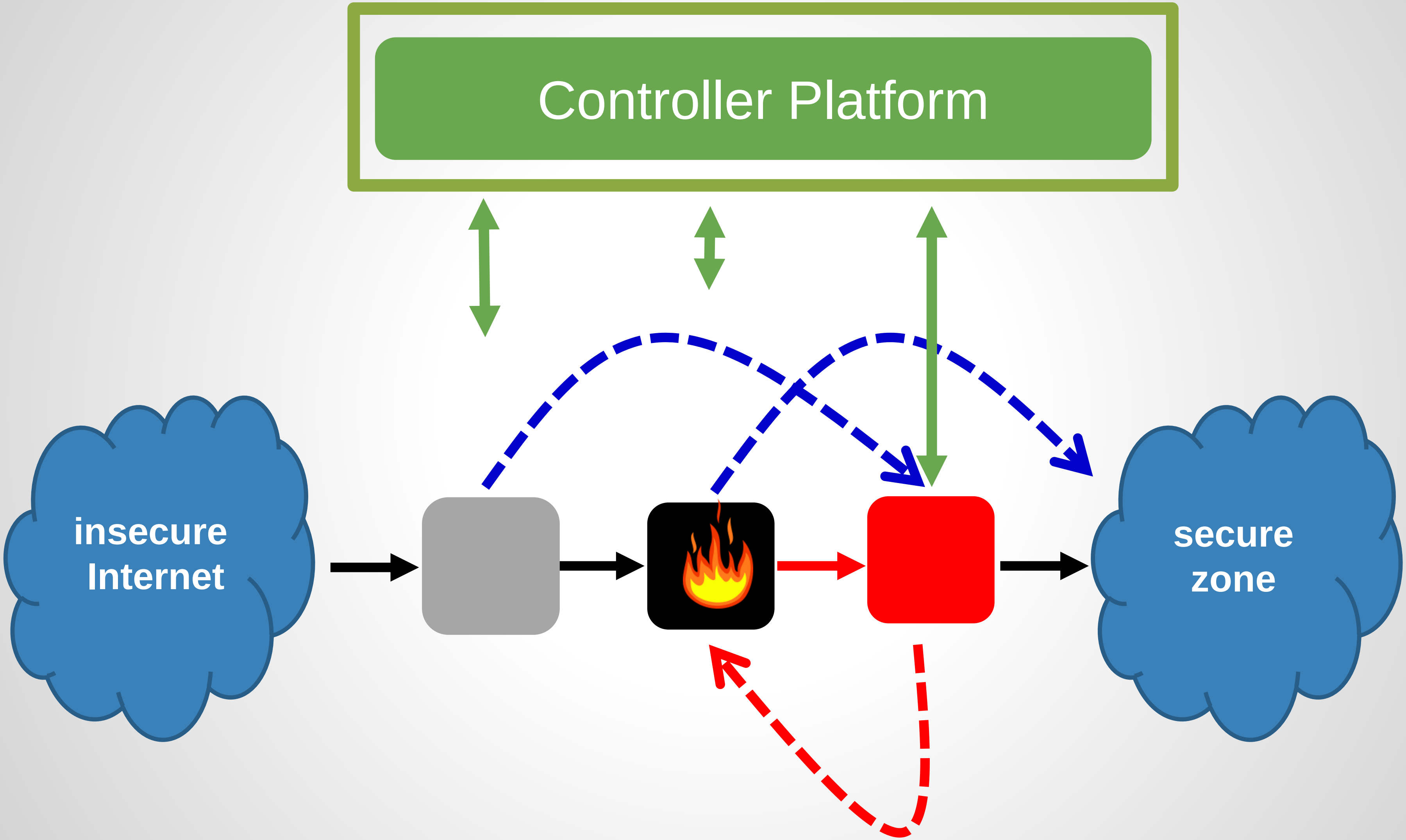
What Can Go Wrong?



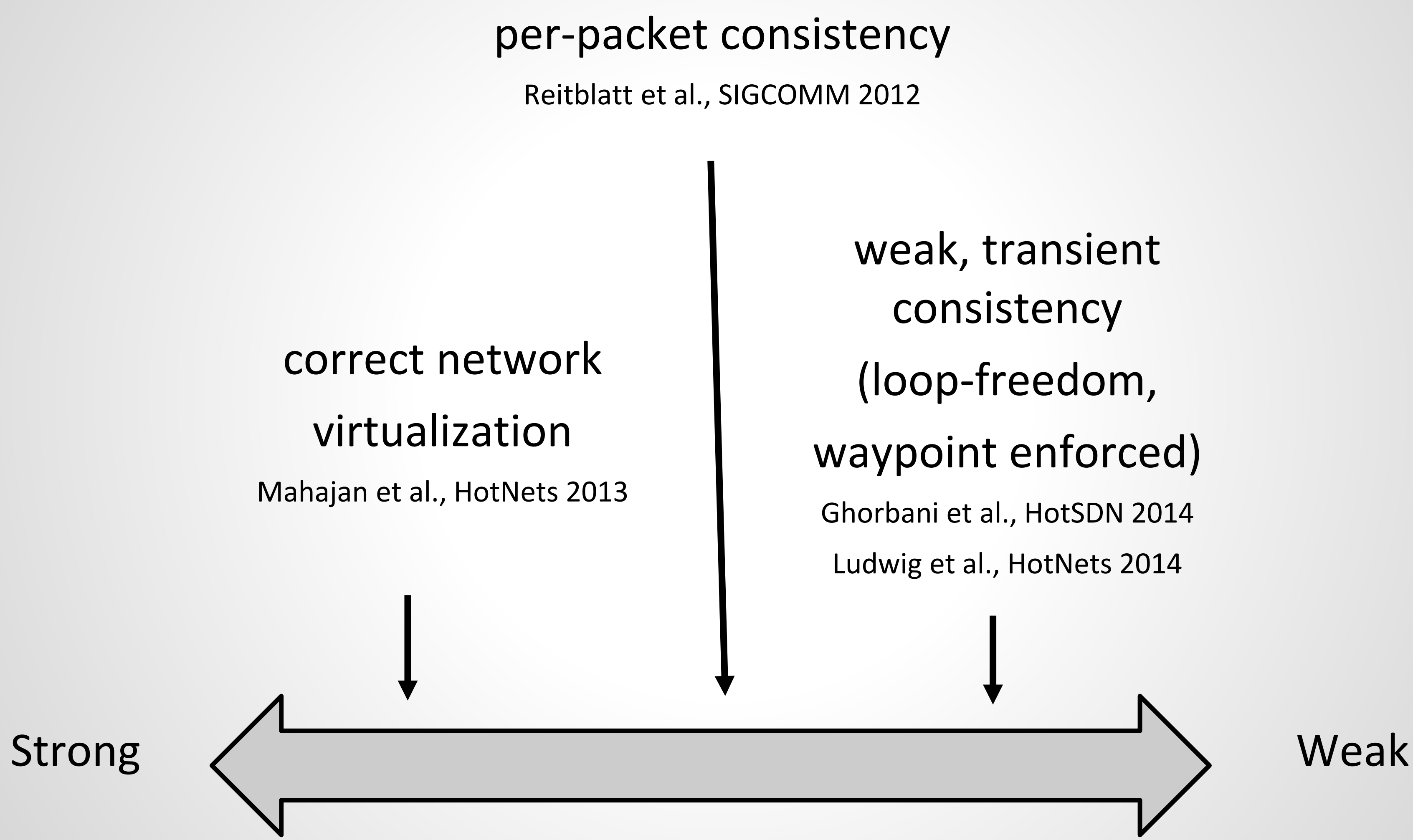
Example 1: Bypassed Waypoint



Example 2: Loops



The Spectrum of Consistency

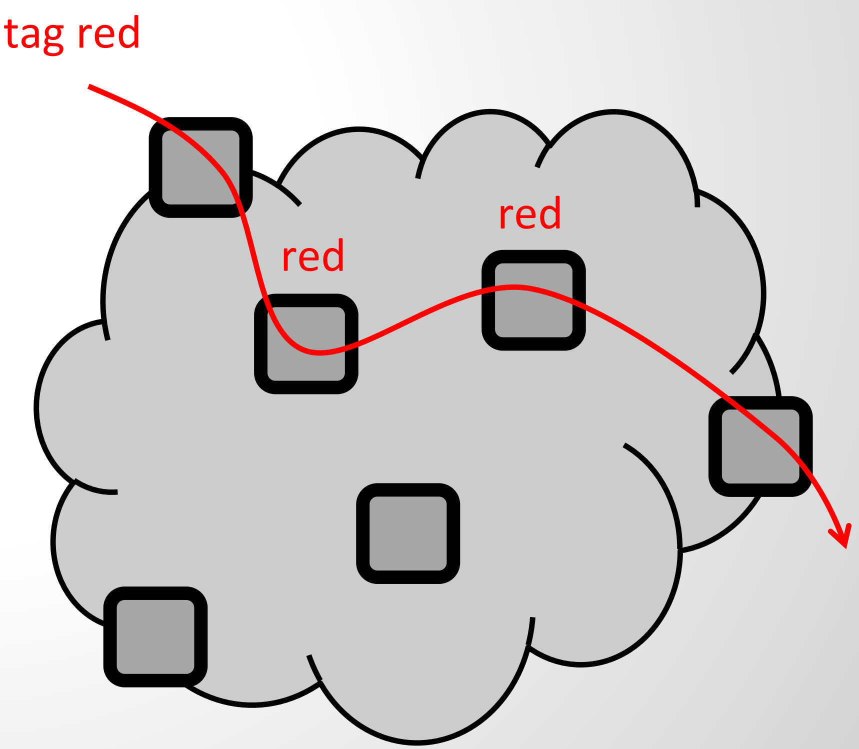


Example: Per-Packet Consistency

Definition: Any packet should either traverse the old route, or the new route, but not a mixture

Implementation:

- 2-Phase Installation
- Tagging at ingress port

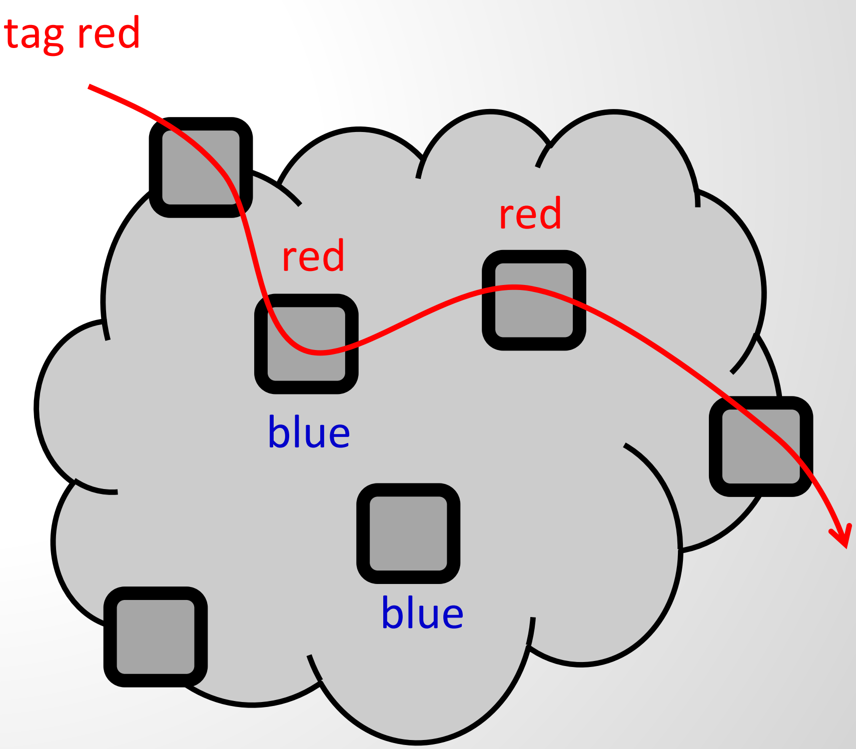


Example: Per-Packet Consistency

Definition: Any packet should either traverse the old route, or the new route, but not a mixture

Implementation:

- ❑ 2-Phase Installation
- ❑ Tagging at ingress port

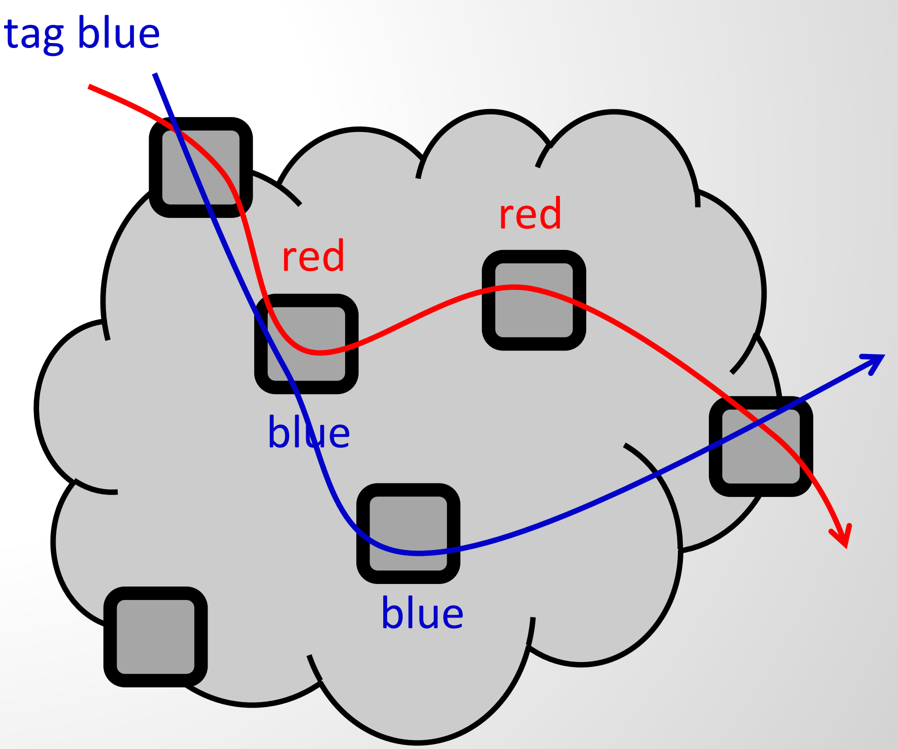


Example: Per-Packet Consistency

Definition: Any packet should either traverse the old route, or the new route, but not a mixture

Implementation:

- ❑ 2-Phase Installation
- ❑ Tagging at ingress port

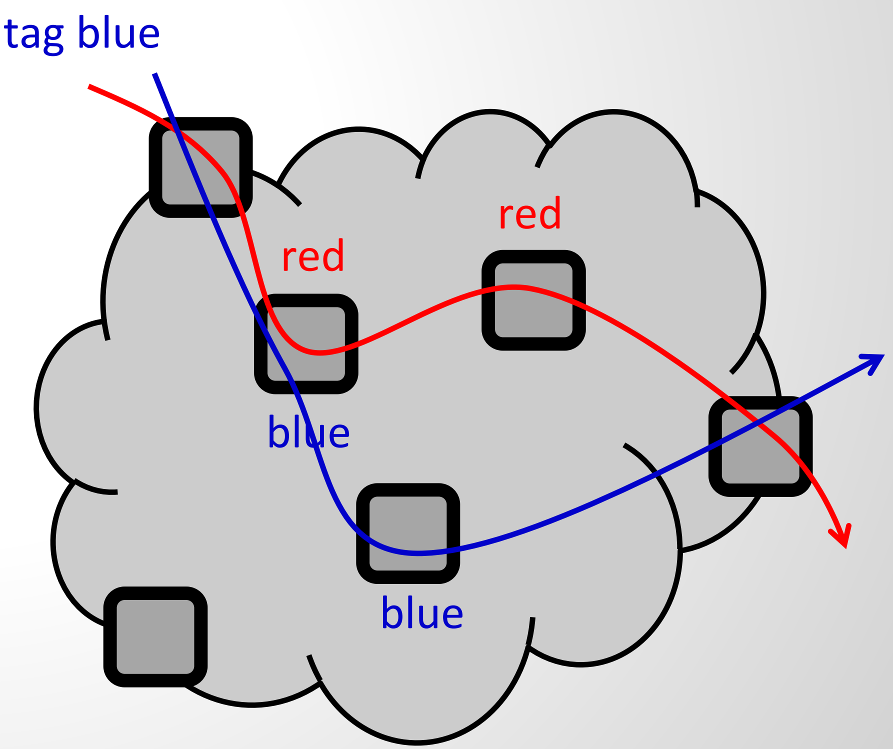


Example: Per-Packet Consistency

Definition: Any packet should either traverse the old route, or the new route, but not a mixture

Implementation:

- 2-Phase Installation
- Tagging at ingress port

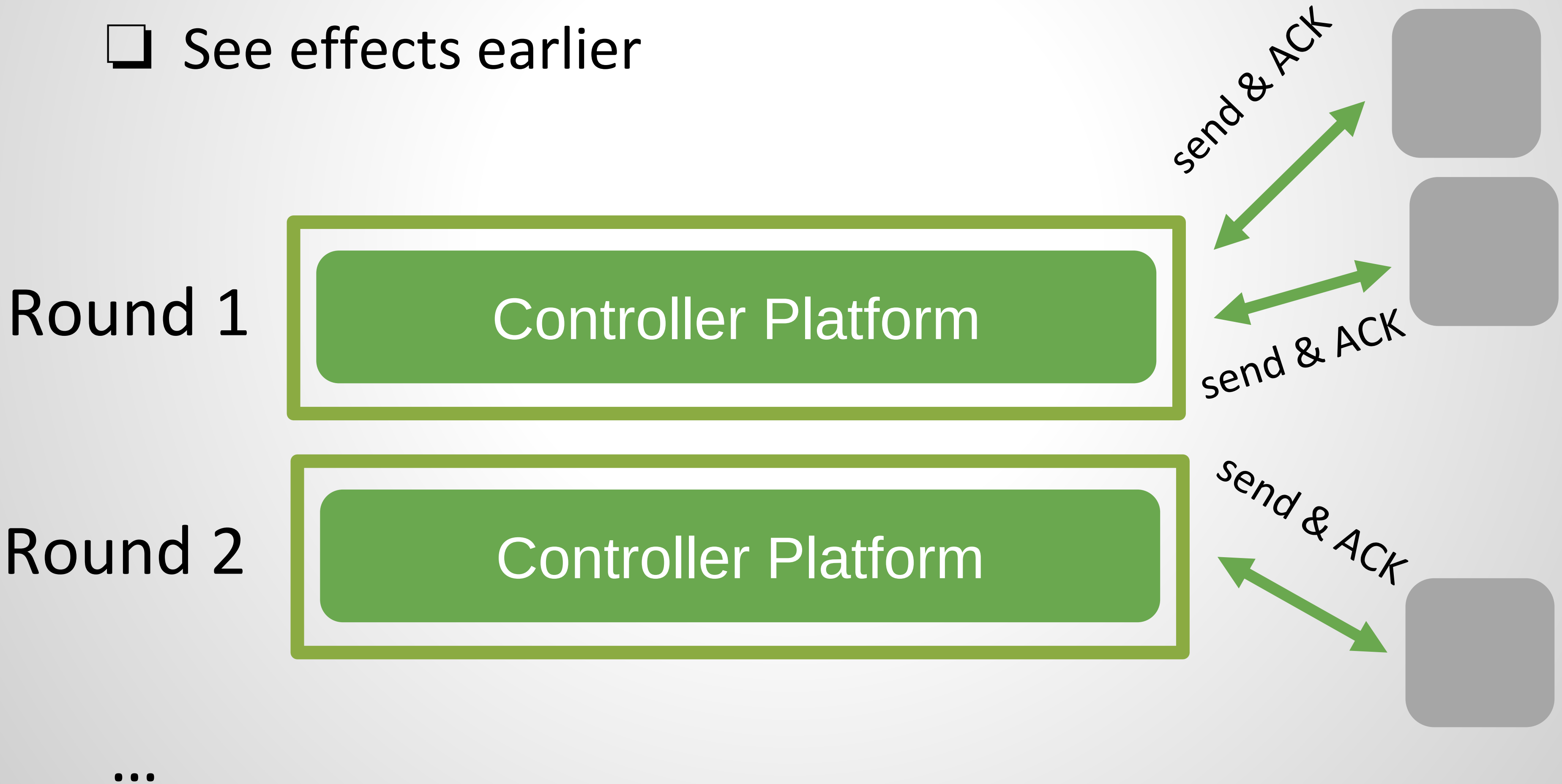


Disadvantages:

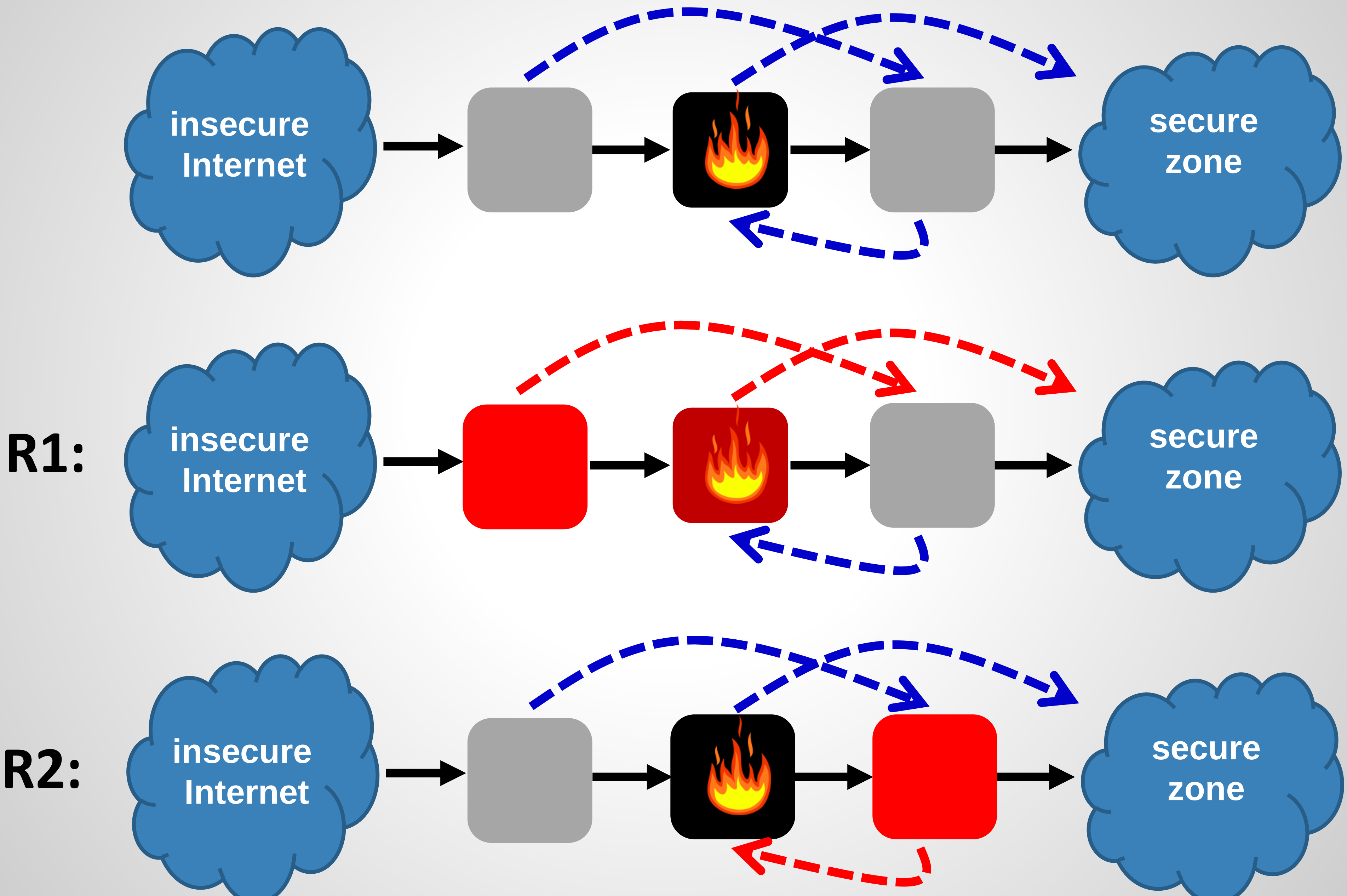
- Tagging: memory
- Delayed effects

Implementing weaker transient consistency?

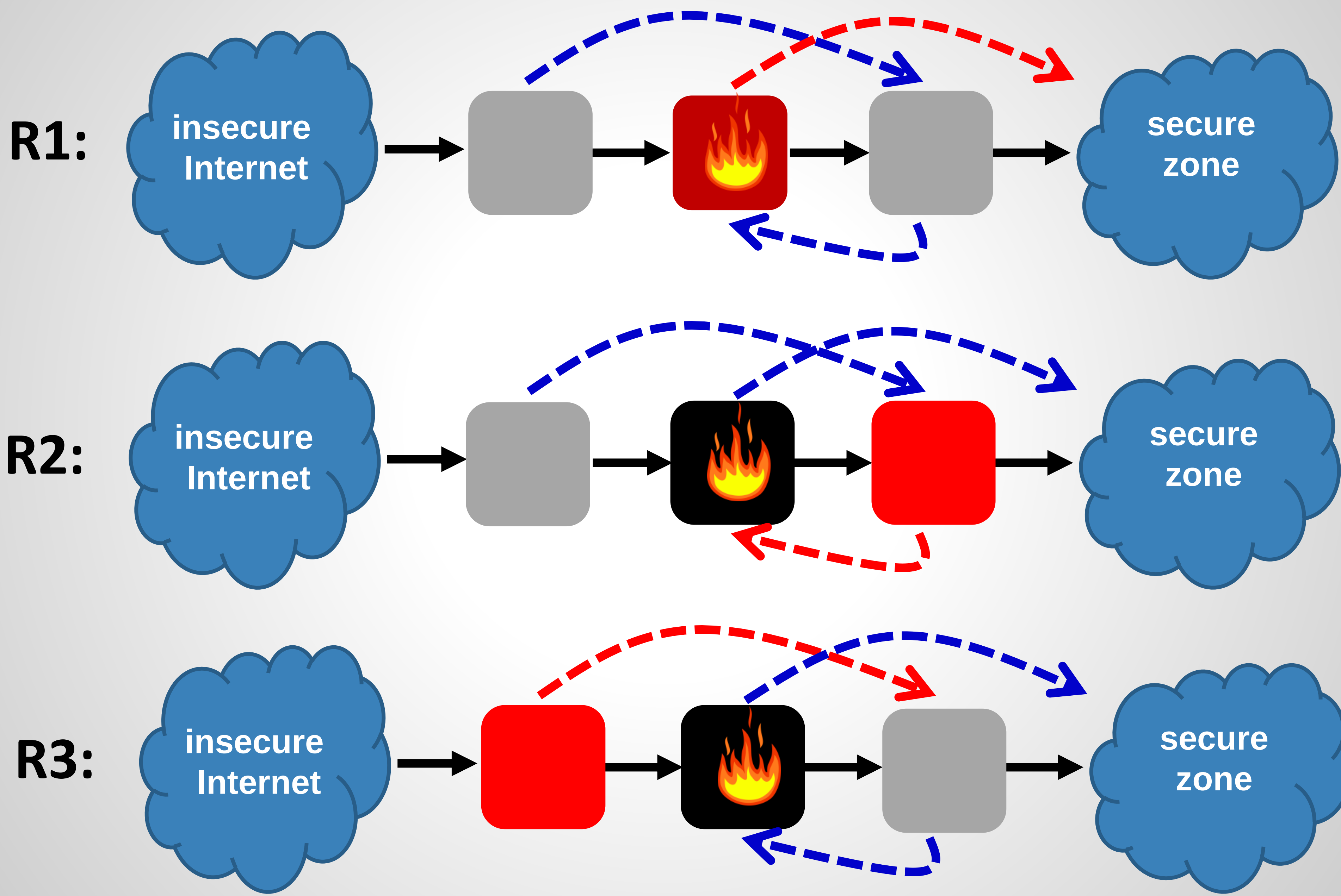
- Update in multiple rounds
  - No tagging needed
  - See effects earlier



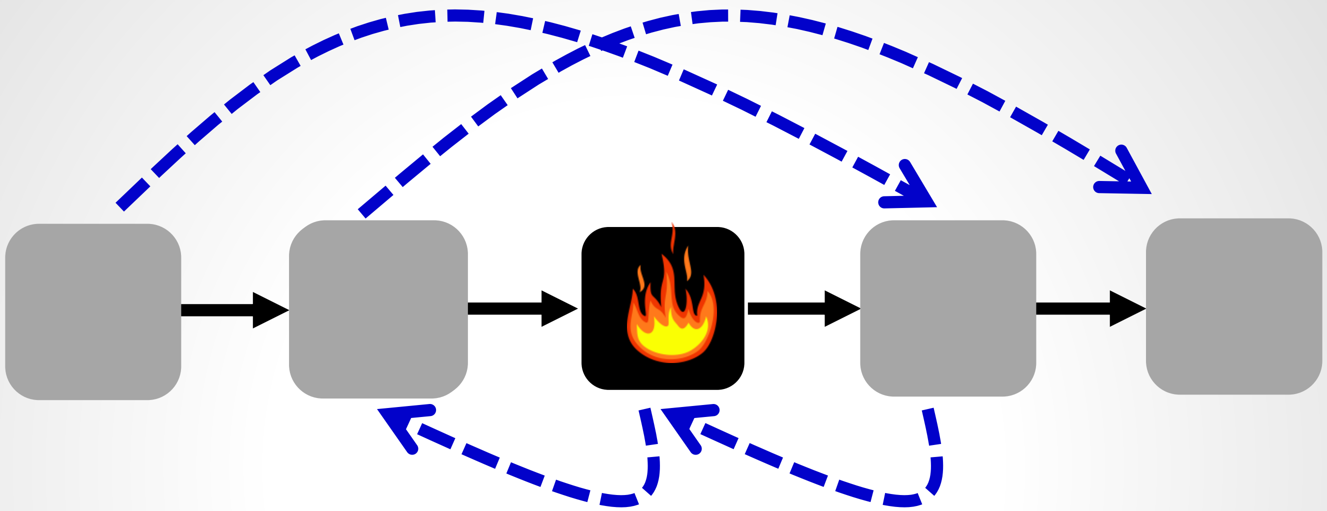
Going Back to Our Examples: LF



Going Back to Our Examples: WPE+LF



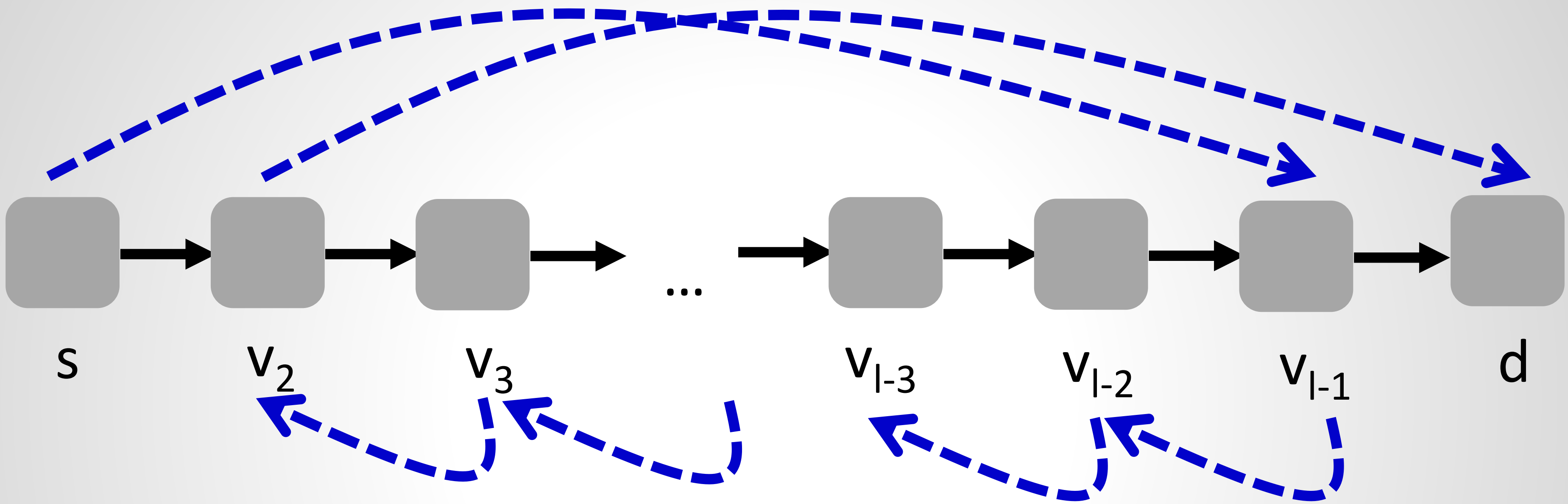
LF and WPE may even conflict!



- ❑ Cannot update forward edge: WP
- ❑ Cannot update backward edge: LF

No schedule exists!

$\Omega(n)$ Rounds



- Must update v_i before v_{i+1}
- Takes $\Omega(n)$ rounds: v_3 v_4 v_5 v_6 …

Reading / Literature Pointer

- A nice talk by Jennifer Rexford:
  http://materials.dagstuhl.de/files/15/15071/15071.JenniferRexford.Slides.pptx

- 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.

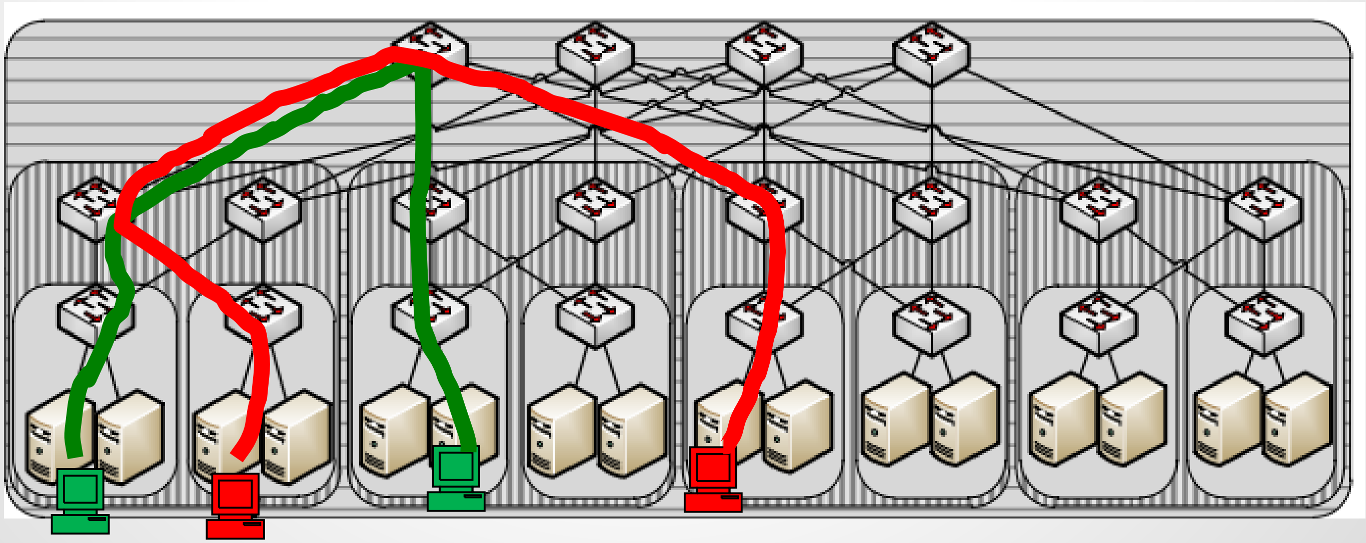
- Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies
  Arne Ludwig, Matthias Rost, Damien Foucard, and Stefan Schmid.
  13th ACM Workshop on Hot Topics in Networks (**HotNets**), Los Angeles, California, USA, October 2014.

Rough Plan

- ❑ SDN and Network Virtualization: Debunking some myths and misunderstandings
- ❑ **Some fundamental research challenges**
- ❑ Mini-Tutorial: How to exploit flexiblities in virtualized networked systems?
- ❑ Mini-Tutorial: How are datacenters designed?
- ❑ Mini-Tutorial: Put your hands on SDN

Cloud Computing + Networking?
Network matters!

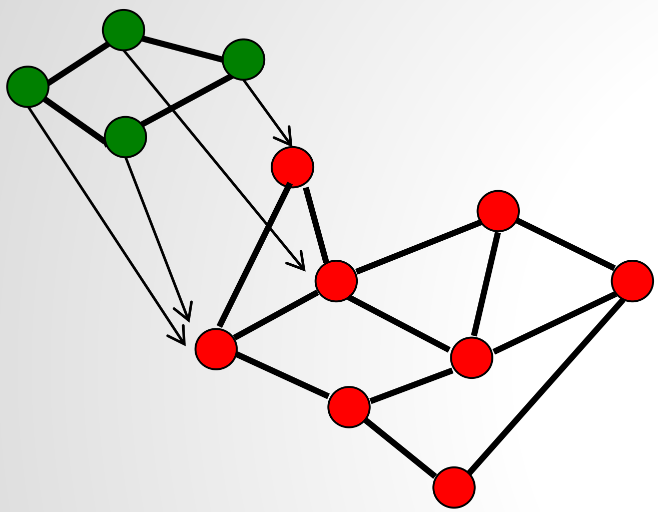
- Example: Batch Processing Applications such as Hadoop
  - **Communication intensive**: e.g., shuffle phase
  - Example Facebook: 33% of **execution time** due to communication
- For predictable preformance in shared cloud: need explicit bandwidth reservations!



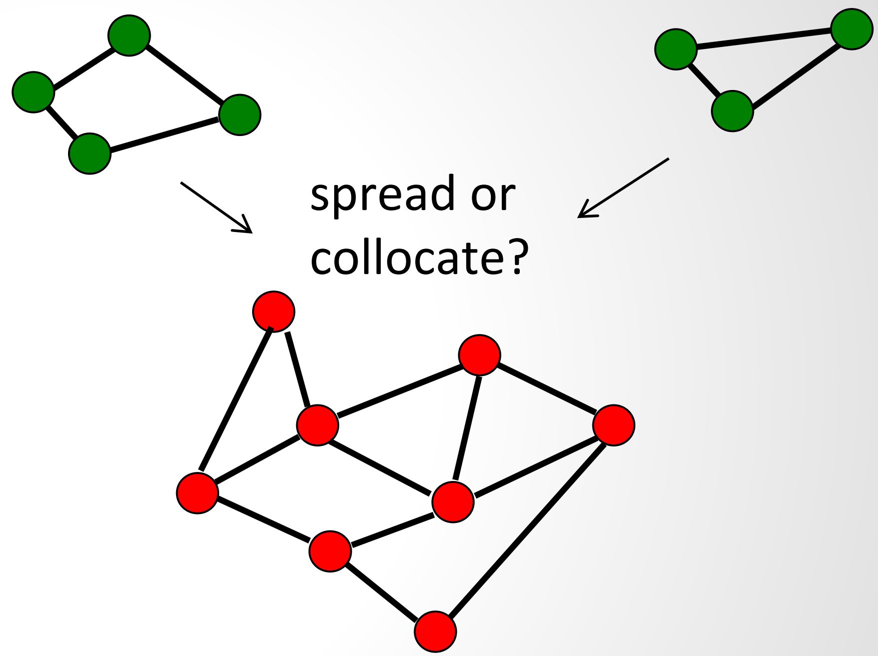
- How to max utilization? A **network embeddig problem**!

Flavors of VNet Embedding Problems (VNEP)

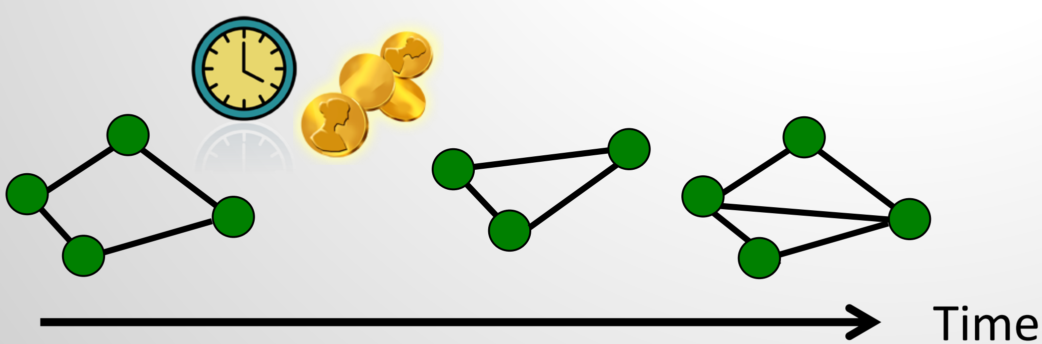
Minimize embedding footprint of a single VNet :



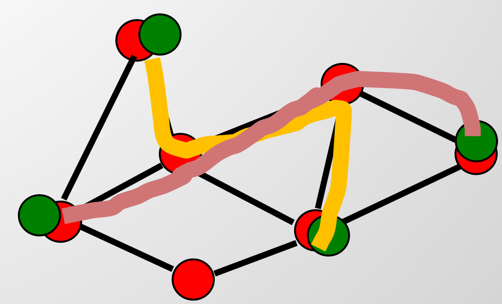
Minimize max load of multiple VNets or collocate to save energy:



Maximize profit over time:



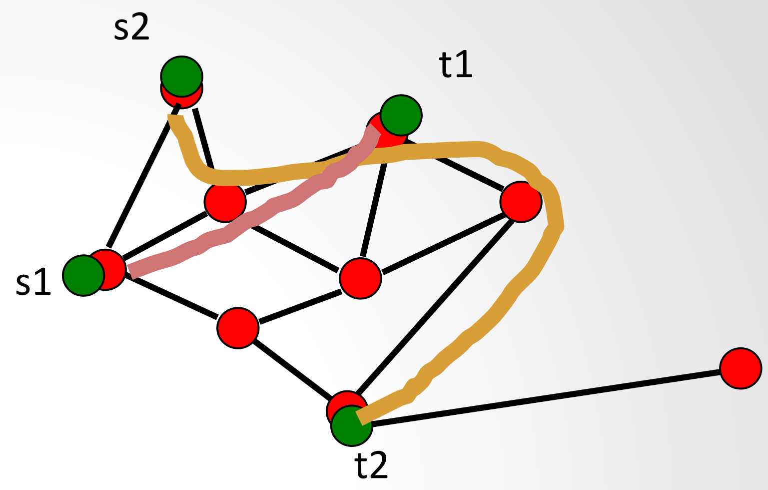
Endpoints fixed:



Let’s Exploit Allocation Flexibilities to Maximize Utilization

Let’s Exploit Allocation Flexibilities to Maximize Utilization

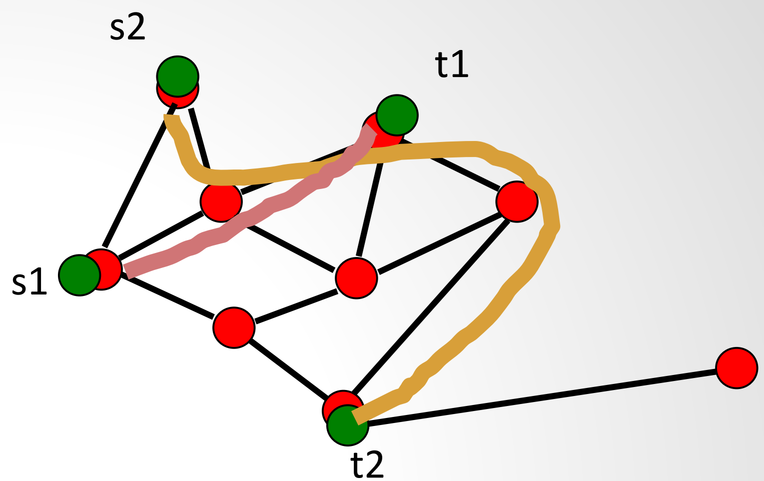
Start simple: exploit flexible routing between given VMs



Let's Exploit Allocation Flexibilities to Maximize Utilization

Start simple: exploit flexible routing between given VMs

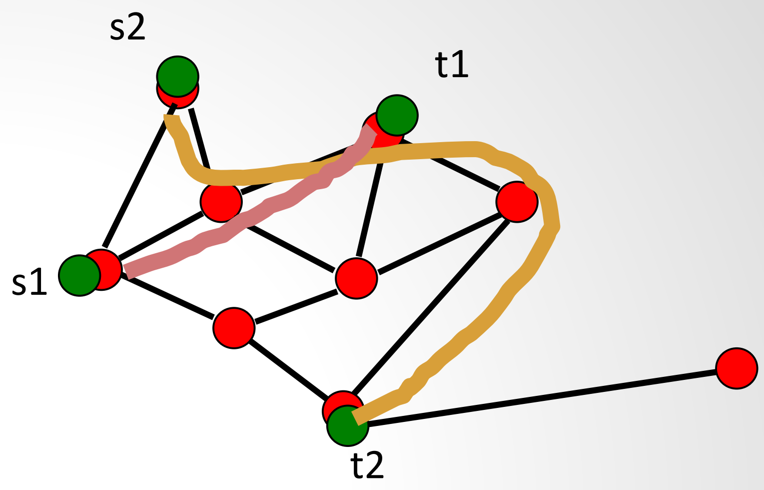
- Integer multi-commodity flow problem with 2 flows?



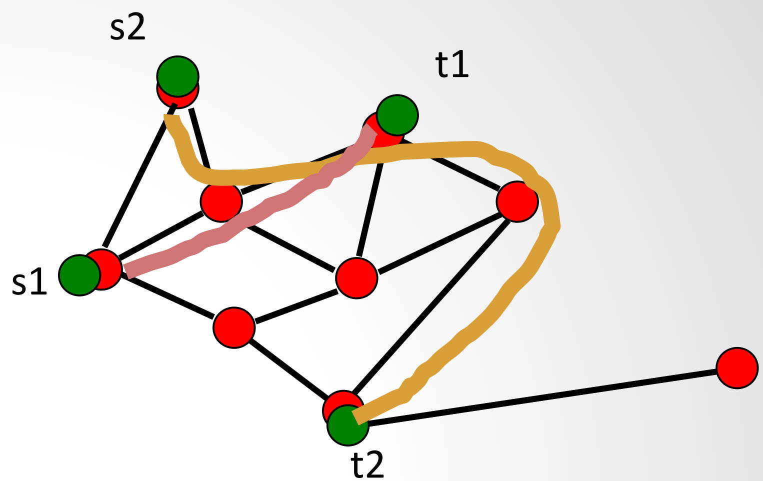
Let’s Exploit Allocation Flexibilities to Maximize Utilization

Start simple: exploit flexible routing between given VMs

- Integer multi-commodity flow problem with 2 flows?
- Oops: NP-hard

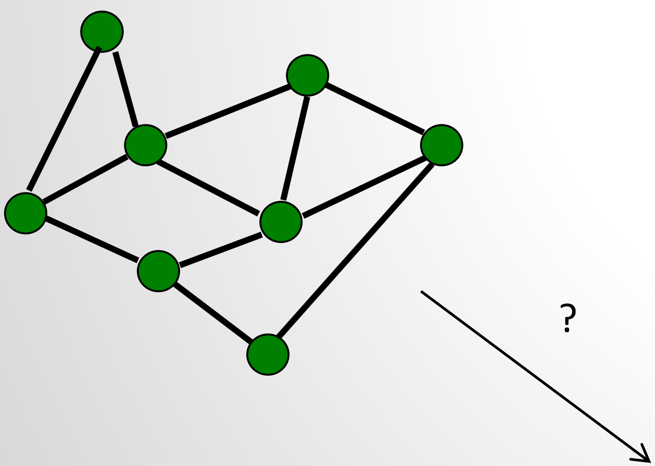


Let's Exploit Allocation Flexibilities to Maximize Utilization



Start simple: exploit flexible routing between given VMs

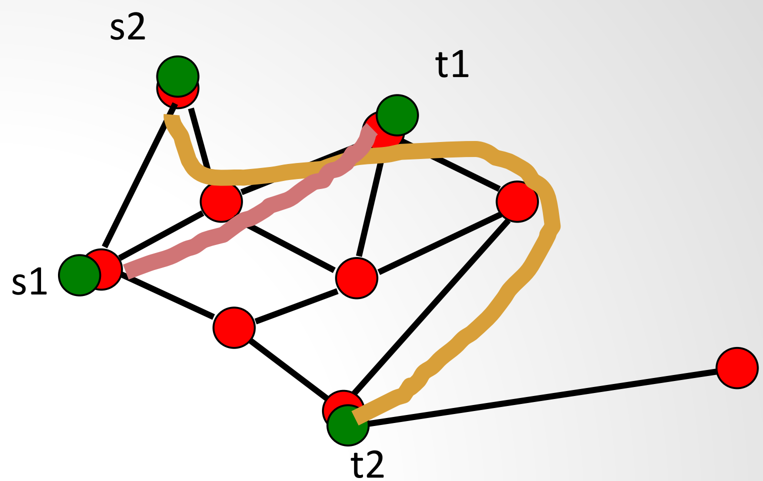
- Integer multi-commodity flow problem with 2 flows?
- Oops: NP-hard



Forget about paths: exploit VM placement flexibilities!

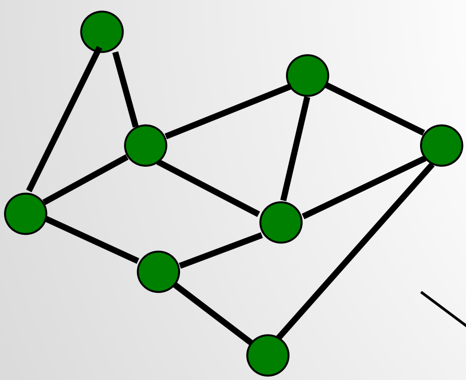
- Most simple: Minimum Linear Arrangement without capacities

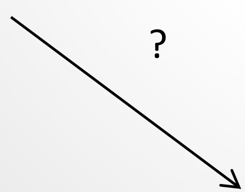
Let’s Exploit Allocation Flexibilities to Maximize Utilization



Start simple: exploit flexible routing between given VMs

- Integer multi-commodity flow problem with 2 flows?
- Oops: NP-hard





Forget about paths: exploit VM placement flexibilities!

- Most simple: Minimum Linear Arrangement without capacities
- NP-hard ☹


That's all Folks!

Wait a minute!

These problems need to be solved!

And they often can, even with guarantees.

Rough Plan

- ❑ SDN and Network Virtualization: Debunking some myths and misunderstandings
- ❑ Some fundamental research challenges
- ❑ **Mini-Tutorial: How to exploit flexiblities in virtualized networked systems?**
- ❑ Mini-Tutorial: How are datacenters designed?
- ❑ Mini-Tutorial: Put your hands on SDN

Solving the VNEP

- **Formulate a Mixed Integer Program**
- Leverage additional structure
- Use online primal-dual approach

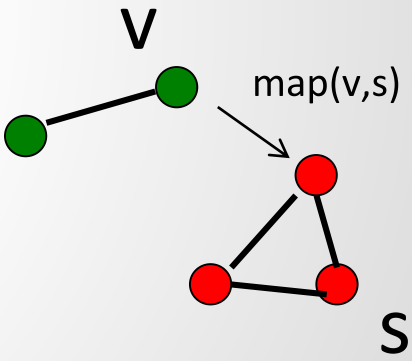
Mixed Integer Programs (1)

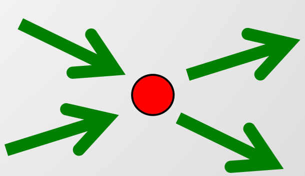
Recipe for VNEP formulation :

- A (linear) objective function (e.g., load or footprint)
- A set of (linear) constraints
- Feed it to your favorite solver (CPLEX, Gurobi, etc.)

Details:

- Introduce binary variables ***map(v,s)*** to map virtual nodes *v* on substrate node *s*
- Introduce flow variables for paths (splittable or not?)
- Ensure **flow conservation**: all flow entering a node must leave the node, unless it is the source or the destination





Mixed Integer Programs (1)

Constants:

Requests : R

Substrate Vertices : V_s

Substrate Edges : $E_s : V_s \times V_s$

Unique : $uni\_check_s : \forall(s_1, s_2) \in E_s : (s_2, s_1) \notin E_s$

SNode Capacity : $snc(s) \rightarrow \mathbb{R}^+, s \in V_s$

SLink Capacity : $slc(e_s) \rightarrow \mathbb{R}^+, e_s \in E_S$

Virtual Vertices : $V_v(r), r \in R$

Virtual Edges : $E_v(r) :\rightarrow V_v(r) \times V_v(r), r \in R$

Unique : $uni\_check_v : \forall r \in R, (v_1, v_2) \in E_v(r) : (v_2, v_1) \notin E_v(r)$

VNode Demand : $vnd(r, v) \rightarrow \mathbb{R}^+, r \in R, v \in V_v(r)$

VEdge Demand : $vld(r, e_v) \rightarrow \mathbb{R}^+, r \in R, e_v \in E_v(r)$

Edges-Reverse : $ER_s : \forall(s_1, s_2) \in E_s \exists (s_2, s_1) \in ER_s \wedge |E_s| = |ER_s|$

Edges-Bidirectional : $EB_s : E_s \cup ER_s$

Migration Cost : $mig\_cost(r, v, s) \rightarrow \mathbb{R}^{+\ |V_v(r)| \times |V_s|}, r \in R, v \in V_v(r), s \in V_s$

Possible Placements : $place(r, v, s) \rightarrow \{0, 1\}^{|V_v(r)| \times |V_s|}, r \in R, v \in V_v(r), s \in V_s$

Variables:

Node Mapping : $n\_map(r, v, s) \in \{0, 1\}, r \in R, v \in V_v(r), s \in V_s$

Flow Allocation : $f\_alloc(r, e, eb) \geq 0, r \in R, e \in E_v(r), eb \in EB_s$

Constraints:

Each Node Mapped : $\forall r \in R, v \in V_v(r) : \sum_{s \in V_s} n\_map(r, v, s) \cdot place(r, v, s) = 1$

Feasible : $\forall s \in V_s : \sum_{r \in R, v \in V_v(r)} n\_map(r, v, s) \cdot vnd(r, v) \leq snc(s)$

Guarantee Link Realization : $\forall r \in R, (v_1, v_2) \in E_v(r), s \in V_s \sum_{(s, s_2) \in V_s \times V_s \cap EB_s} f\_alloc(r, v_1, v_2, s, s_2) - \sum_{(s_1, s) \in V_s \times V_s \cap EB_s} f\_alloc(r, v_1, v_2, s_1, s) = vld(r, v_1, v_2) \cdot (n\_map(r, v_1, s) - n\_map(r, v_2, s))$

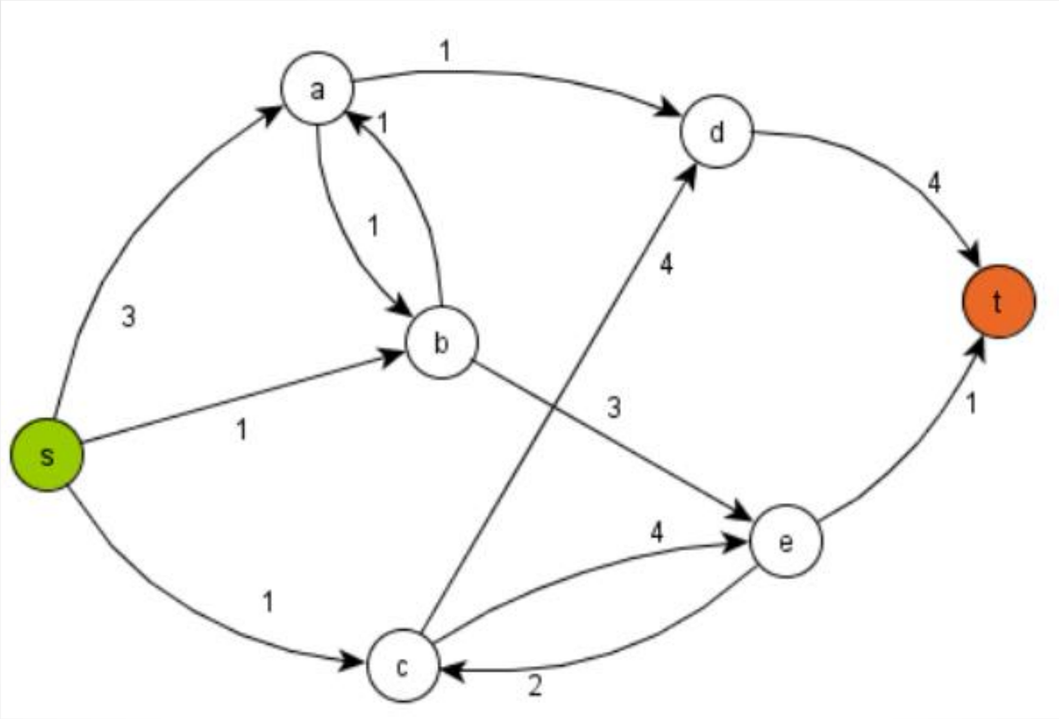
Realize Flows : $\forall (s_1, s_2) \in E_s \sum_{r \in R, (v_1, v_2)} f\_alloc(r, v_1, v_2, s_1, s_2) + f\_alloc(r, v_1, v_2, s_2, s_1) \leq slc(s_1, s_2)$

Objective function:

Minimize Embedding Cost : $min : \sum_{r \in R, (v_1, v_2) \in E_v(r), (s_1, s_2) \in E_s} f\_alloc(r, v_1, v_2, s_1, s_2) + f\_alloc(r, v_1, v_2, s_2, s_1)$

entering a node must leave the node,
unless it is the source or the destination

Example: Flow Conservation (1)



Except for source s and destination t,

the incoming flow must equal the outgoing flow:

$$\sum_{u:u\to v} f_{uv} = \sum_{w:v\to w} f_{vw}, \quad \forall v \neq s, t$$

Example: Flow Conservation (2)

But now virtual machines (resp. s and t) are flexible!

Still a linear program?

Example: Flow Conservation (2)

But now virtual machines (resp. s and t) are flexible!

Still a linear program? Yes!

$$\boxed{\forall\ v: \sum_{u} f_{uv} - f_{vu} \geq map(s,v) * b - map(t,v) * \infty}$$

b: constant for requested bandwidth resource from s to t.

map(u,v): binary variable for «is u mapped on v»?

Linear indeed. Cases:

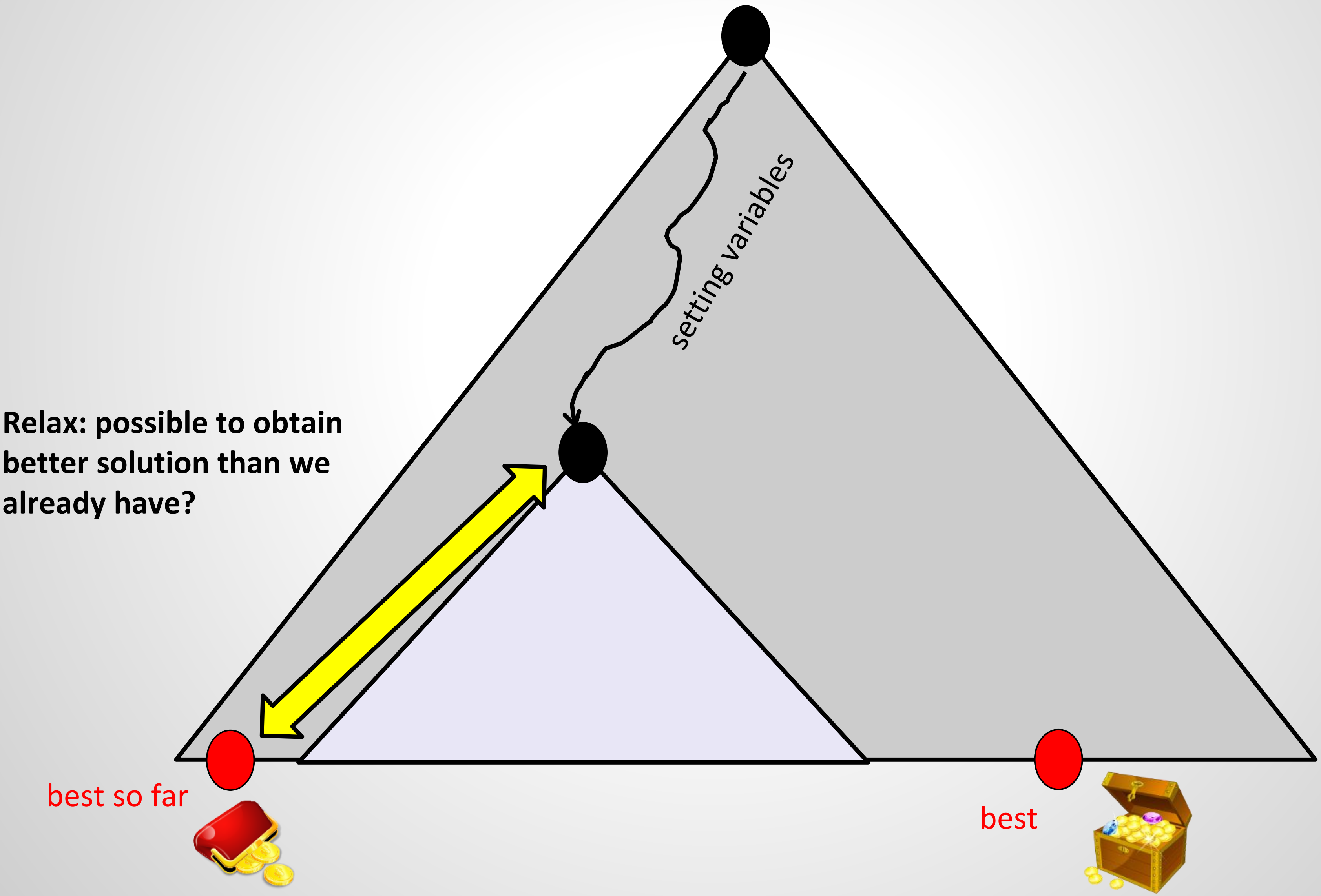
- If v = s ≠ t: outgoing flow at least b
- If v ≠ t ≠ s: RHS 0, flow conservation if =. (≥ handled by objective function)
- If v = t ≠ s: no constraint

Mixed Integer Programs (3)

- MIPs can be quite fast
  - For pure integer programs, SAT solvers likely faster
- However, that's not the end of the story: MIP $\neq$ MIP
  - The specific formulation matters!
- For example: many solvers use relaxations
  - Make integer variables continuous: resulting linear programs (LPs) can be solved in polynomial time!
  - How good can solution in this subtree (given fixed variables) be at most? (More flexibility: solution can only be better!)
  - If already this is worse than currently best solution, we can cut!
- Relaxations can also be used as a basis for heuristics
  - E.g., round fractional solutions to closest integer?

Mixed Integer Programs (4)

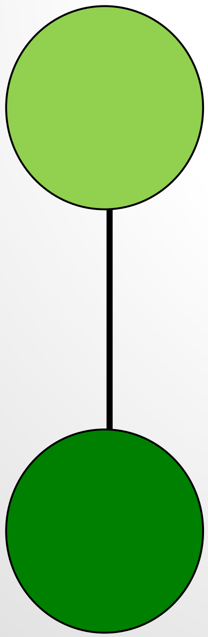
Branch & bound tree:



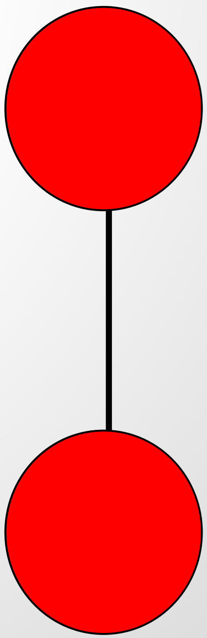
Mixed Integer Programs (5)

- Recall: Relaxations useful if they give good bounds
- However it's hard to formulate a MIP for VNEP which yields useful relaxations!
- What happens here?

VNet:



Physical Network:

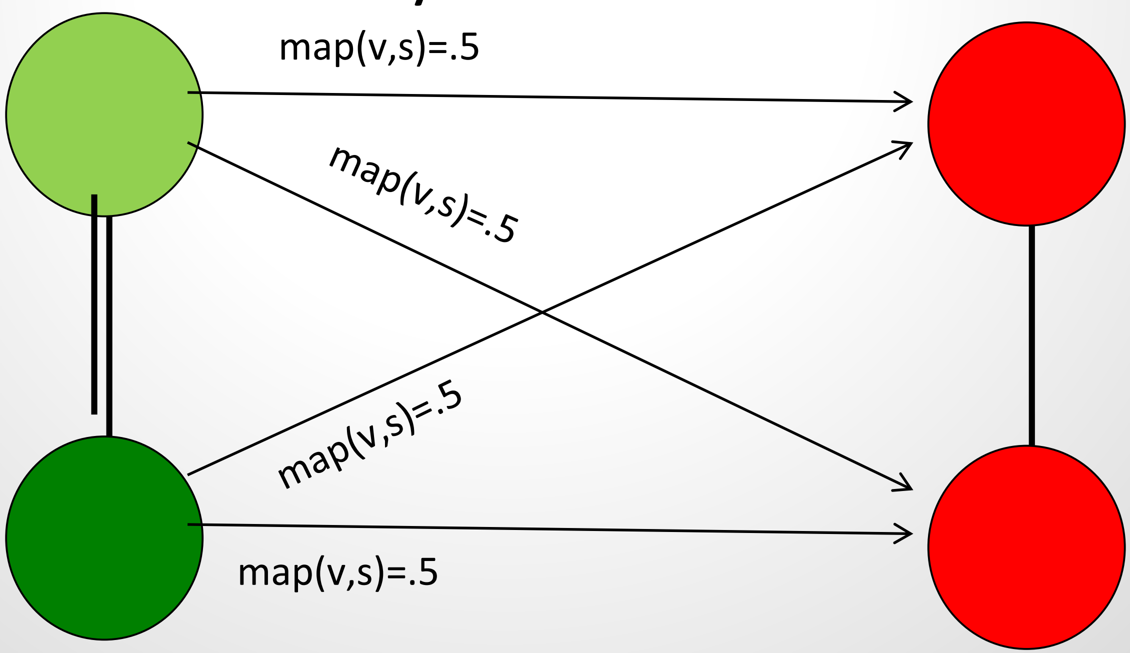


Mixed Integer Programs (5)

- Recall: Relaxations useful if they give good bounds
- However it's hard to formulate a MIP for VNEP which yields useful relaxations!
- What happens here?

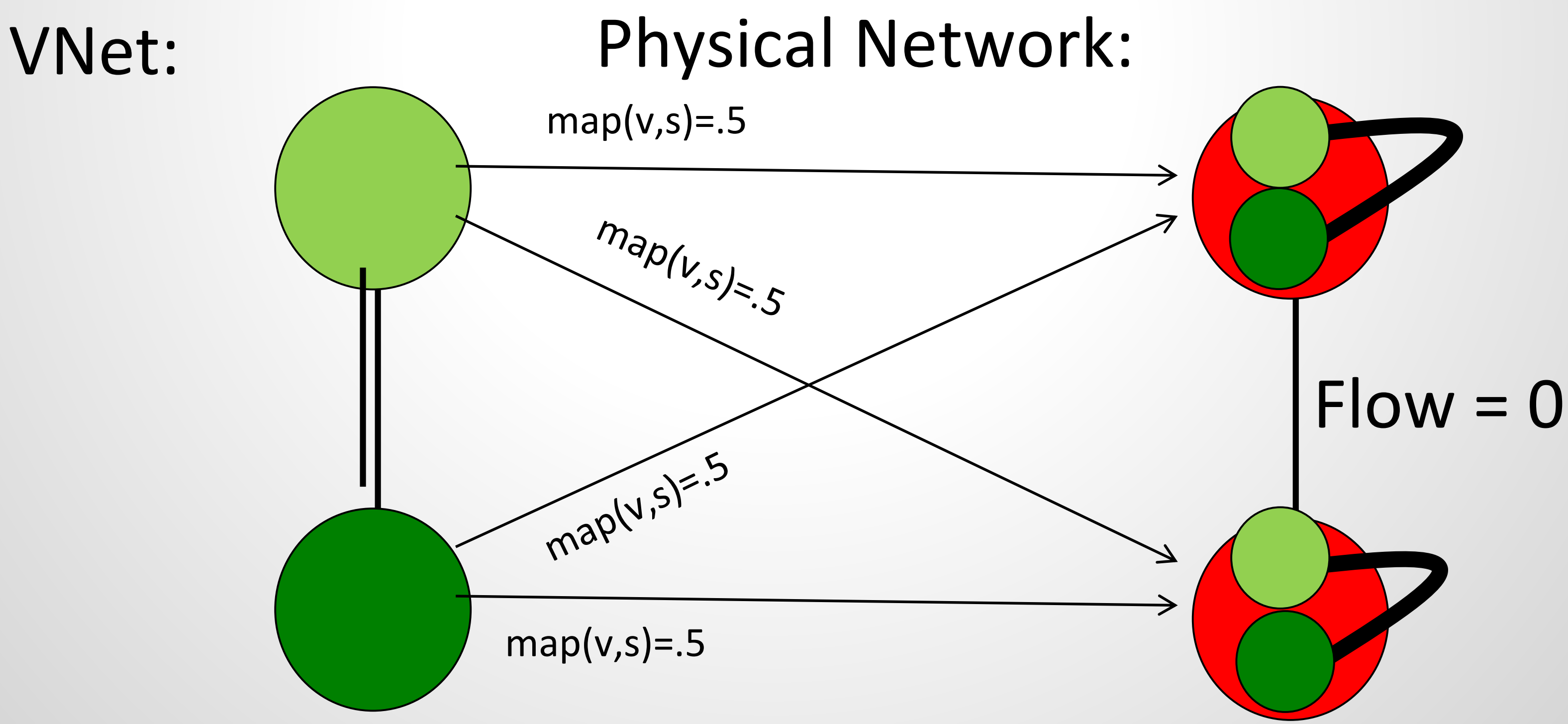
VNet:

Physical Network:



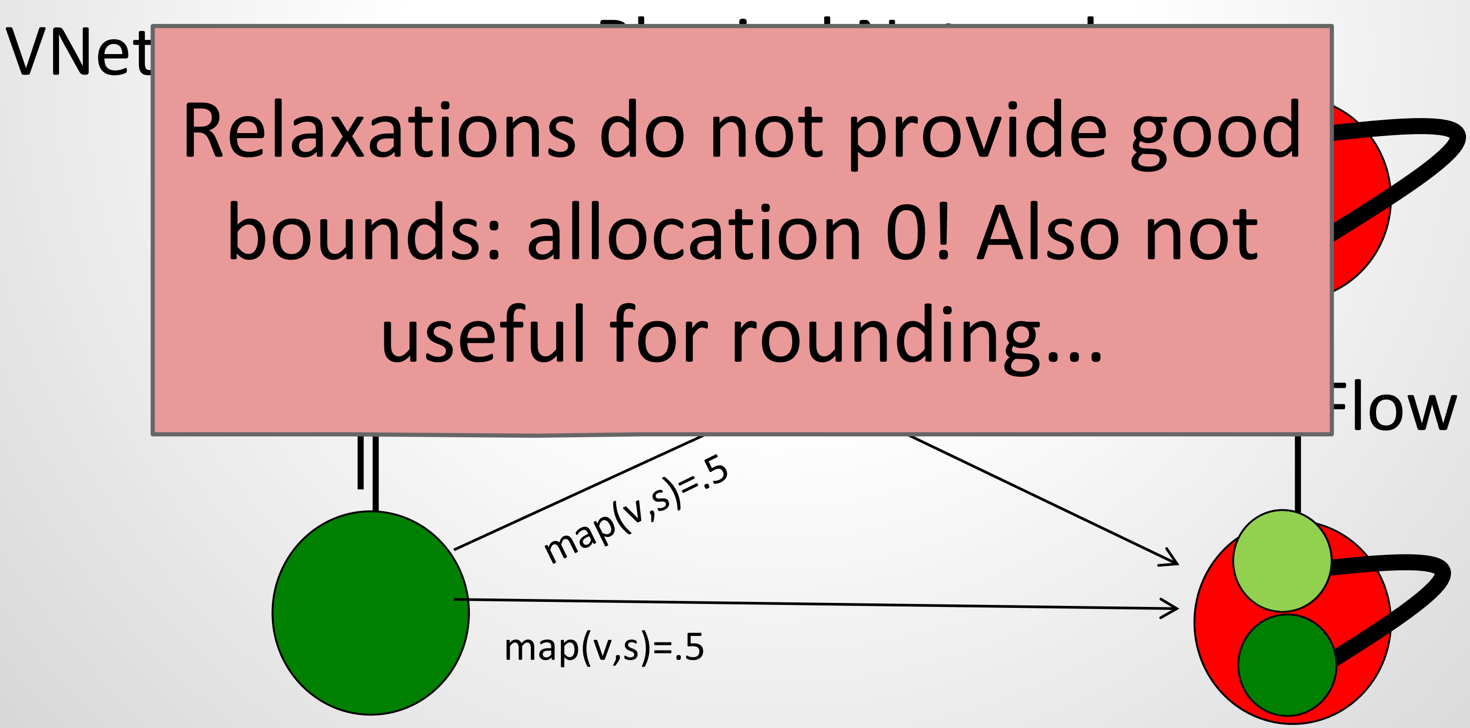
Mixed Integer Programs (5)

- Recall: Relaxations useful if they give good bounds
- However it’s hard to formulate a MIP for VNEP which yields useful relaxations!
- What happens here?



Mixed Integer Programs (5)

- Recall: Relaxations useful if they give good bounds
- However it's hard to formulate a MIP for VNEP which yields useful relaxations!
- What happens here?

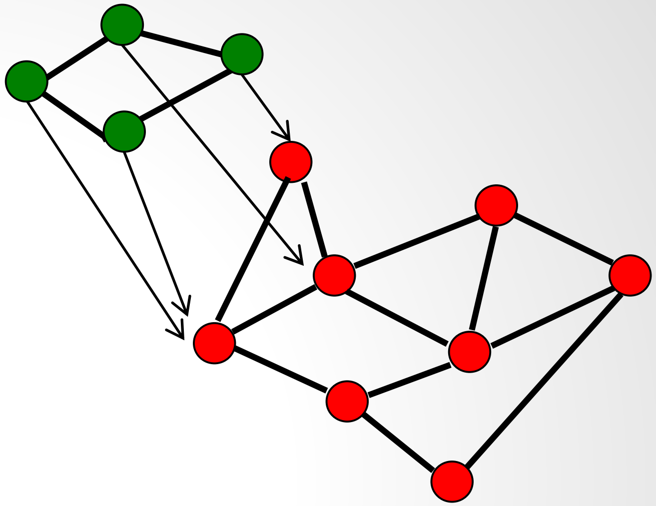


Relaxations do not provide good bounds: allocation 0! Also not useful for rounding...

Solving the VNEP

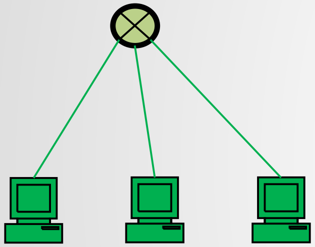
- ❑ Formulate a Mixed Integer Program
- ❑ **Leverage additional structure**
- ❑ Use online primal-dual approach

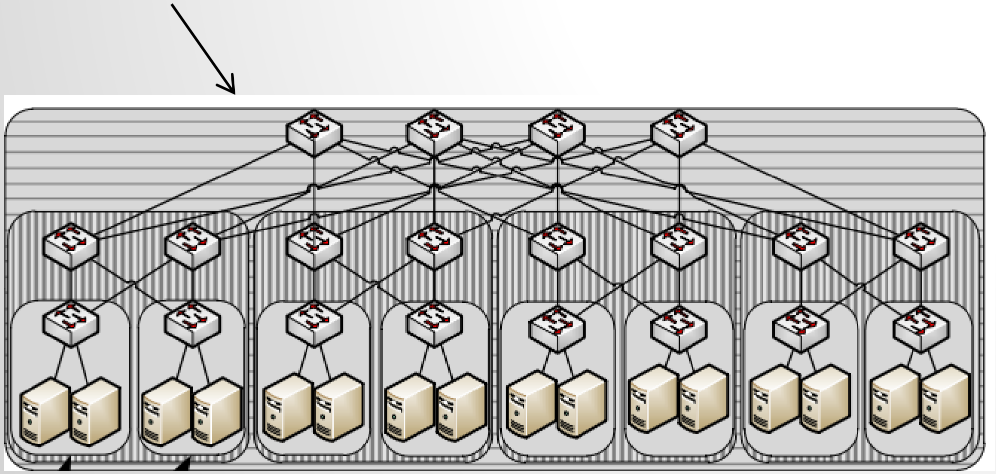
Theory vs Practice



Goal in theory:

Embed as general as possible ***guest graph*** to as general as possible ***host graph***



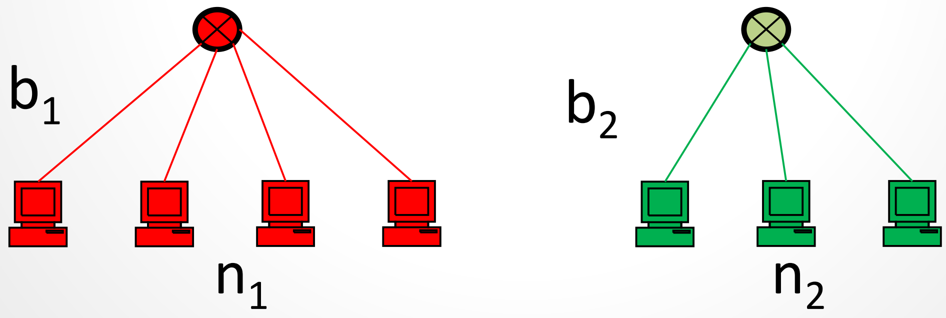


Reality:

Datacenters, WANs, etc. exhibit much **structure** that can be exploited! But also guest networks come with **simple specifications**

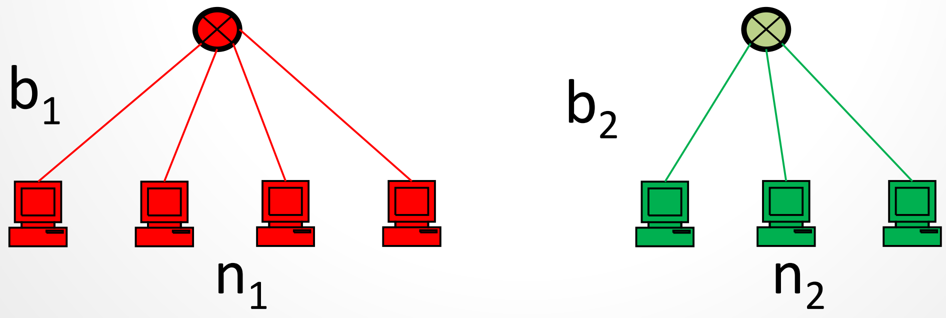
Virtual Clusters

- A prominent abstraction for batch-processing applications: Virtual Cluster *VC(n,b)*
  - Connects *n* virtual machines to a «logical» switch with bandwidth guarantees *b*
  - A simple abstraction



Virtual Clusters

- A prominent abstraction for batch-processing applications: Virtual Cluster *VC(n,b)*
  - Connects *n* virtual machines to a «logical» switch with bandwidth guarantees *b*
  - A simple abstraction

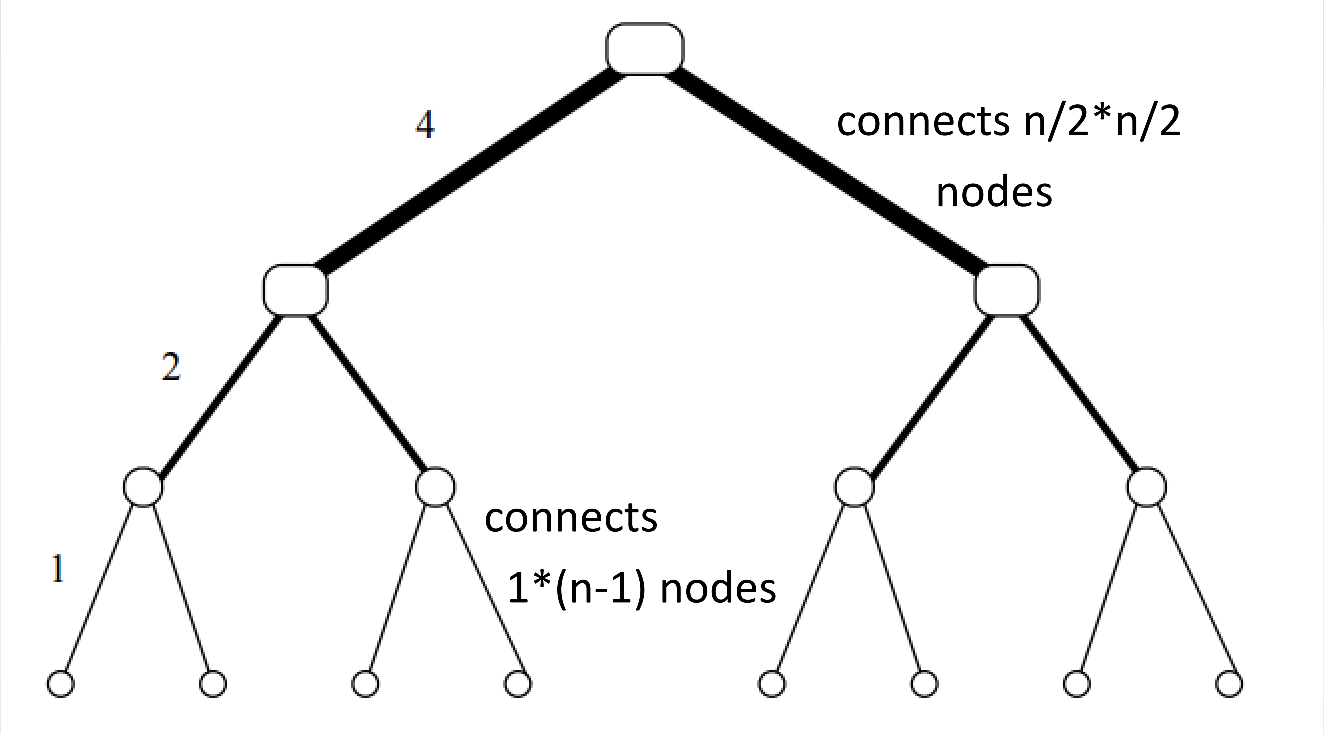


How do datacenter topologies look like?

Rough Plan

- ❑ SDN and Network Virtualization: Debunking some myths and misunderstandings
- ❑ Some fundamental research challenges
- ❑ Mini-Tutorial: How to exploit flexiblities in virtualized networked systems?
- ❑ **Mini-Tutorial: How are datacenters designed?**
- ❑ Mini-Tutorial: Put your hands on SDN

A Typical Datacenter Topology



- ❑ Full bisection bandwidth
- ❑ In practice, often not full bisection

A Brief Tutorial on Datacenter Topologies

Network topologies are often described as graphs!

Graph G=(V,E): V = set of nodes/peers/..., E= set of edges/links/...

d(.,.): distance between two nodes (shortest path), e.g. d(A,D)=?

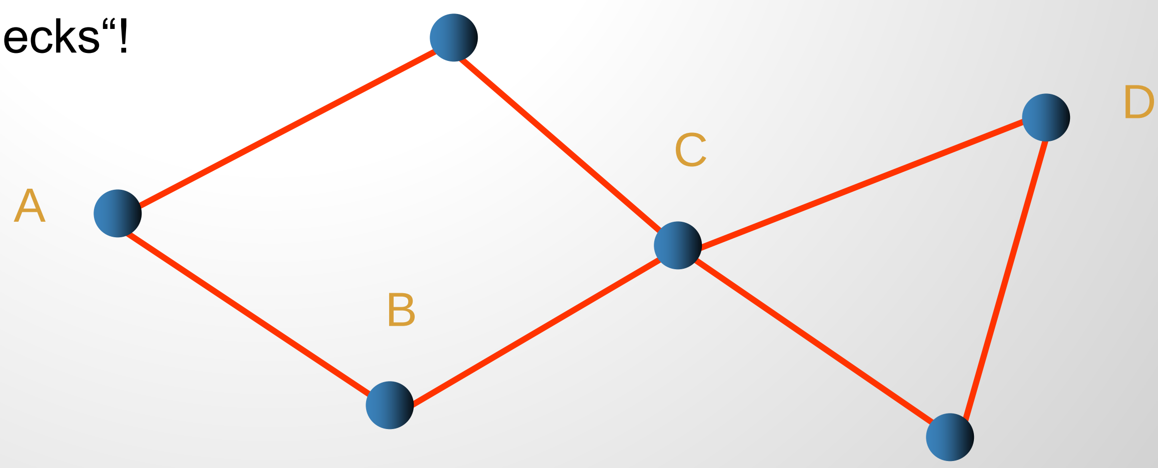
D(G): diameter (D(G)=$\max_{u,v}$ d(u,v)), e.g. D(G)=?

Γ(U): neighbor set of nodes U (not including nodes in U)

α**(U)** = $|\Gamma(U)| / |U|$ (size of neighbor set compared to size of U)

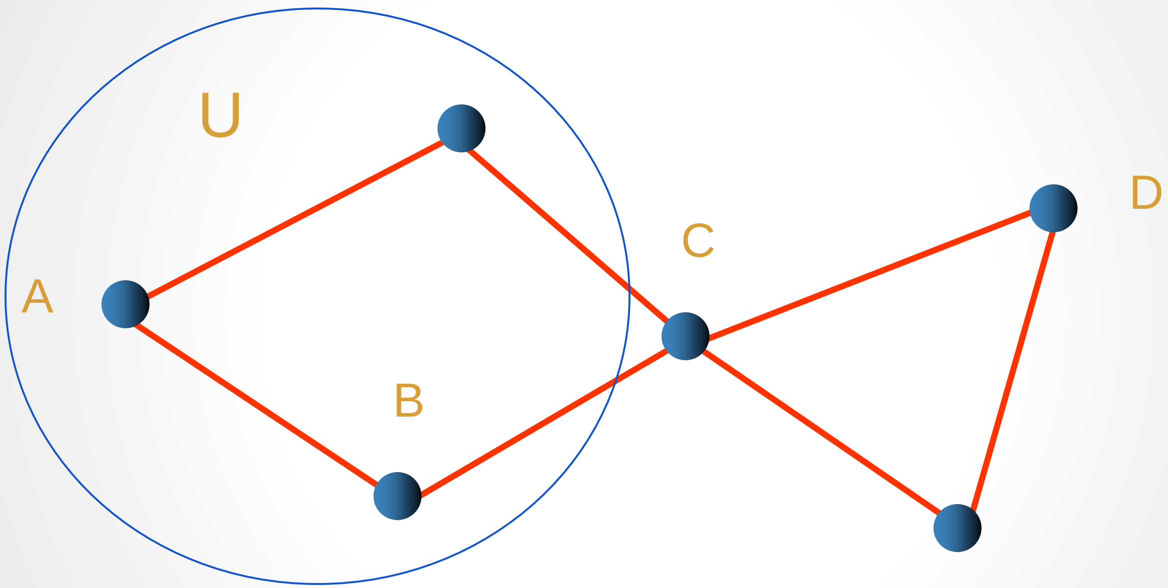
α**(G)** = $\min_{U,\ |U|\cdot\ V/2} \alpha(U)$: expansion of G (meaning?)

Expansion captures „bottlenecks“!



Example

Explanation: Γ(U), α(U)?

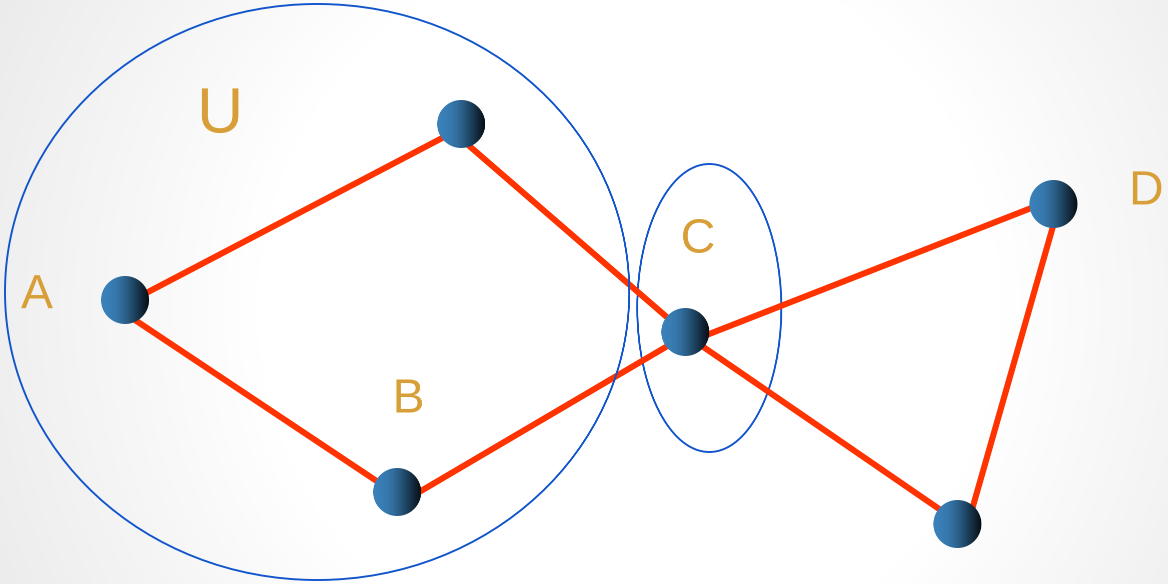


Neighborhood is just {C}, so...

... α=1/3.

Example

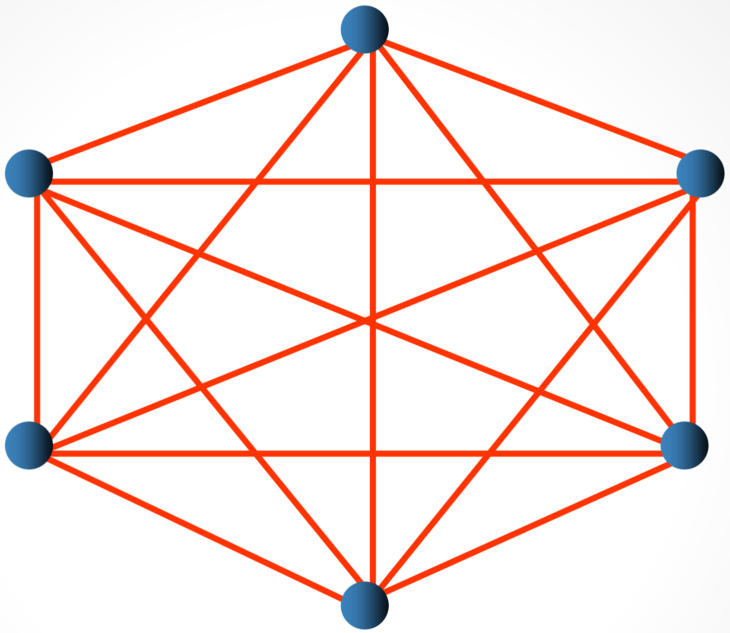
Explanation: Γ(U), α(U)?



α(U)=1/3 (bottleneck!)

What is a good topology?

Complete network: pro and cons?



Pro: robust, easy and fast routing, small diameter...

Cons: does not scale! (degree?, number of edges?, ...)

Why Fat-Tree Networks?

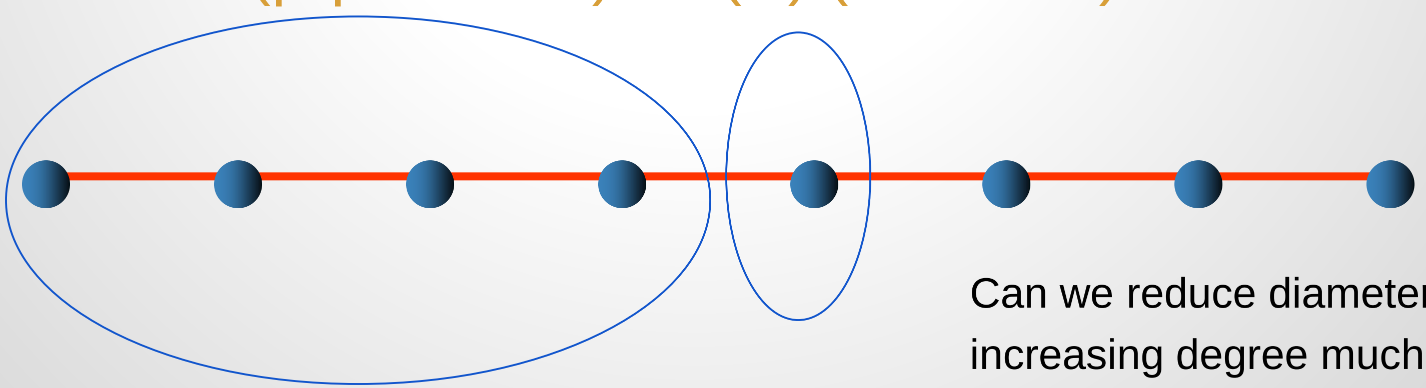
Line network: pro and cons?

Degree? Diameter? Expansion?

Pro: easy and fast routing (tree = unique paths!), small degree (2)...

Cons: does not scale! (diameter = n-1, expansion = 2/n, ...)

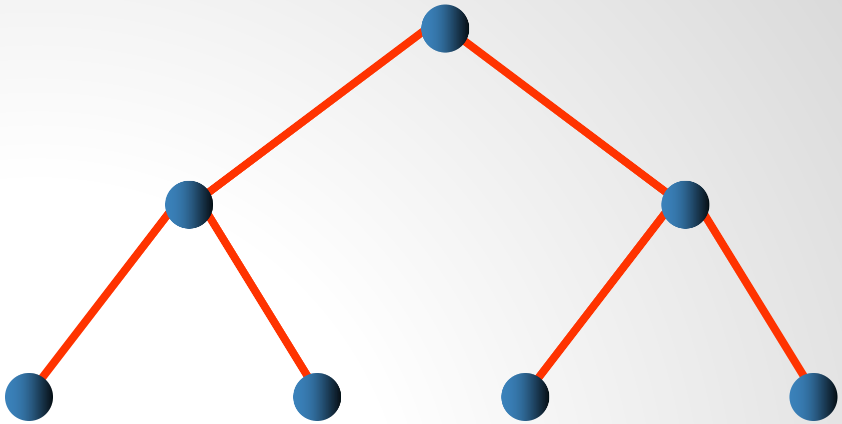
Expansion: U (|V|/2 nodes) Γ(U) (= 1 node)



Can we reduce diameter without increasing degree much?

Why Fat-Tree Networks?

Binary tree network: pro and cons?

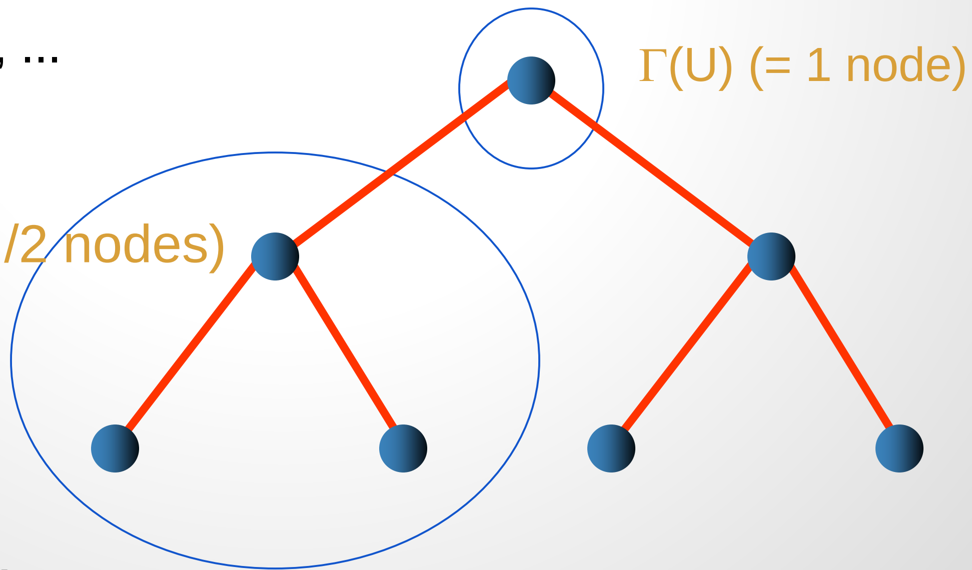


Degree? Diameter? Expansion?

Pro: easy and fast routing (tree = unique paths!), small degree (3), log diameter...

Cons: bad expansion = 2/n, ...

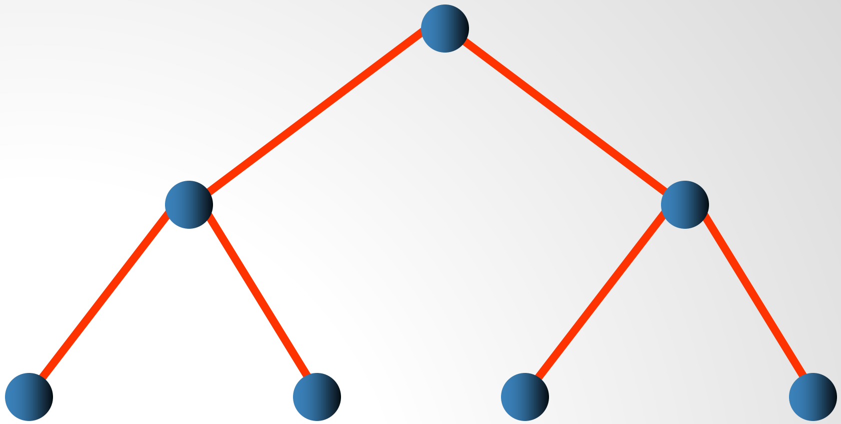
Expansion:



All communication from left to right tree goes through root! ☹ (no «bisection bandwidth»)

Why Fat-Tree Networks?

Binary tree network: pro and cons?

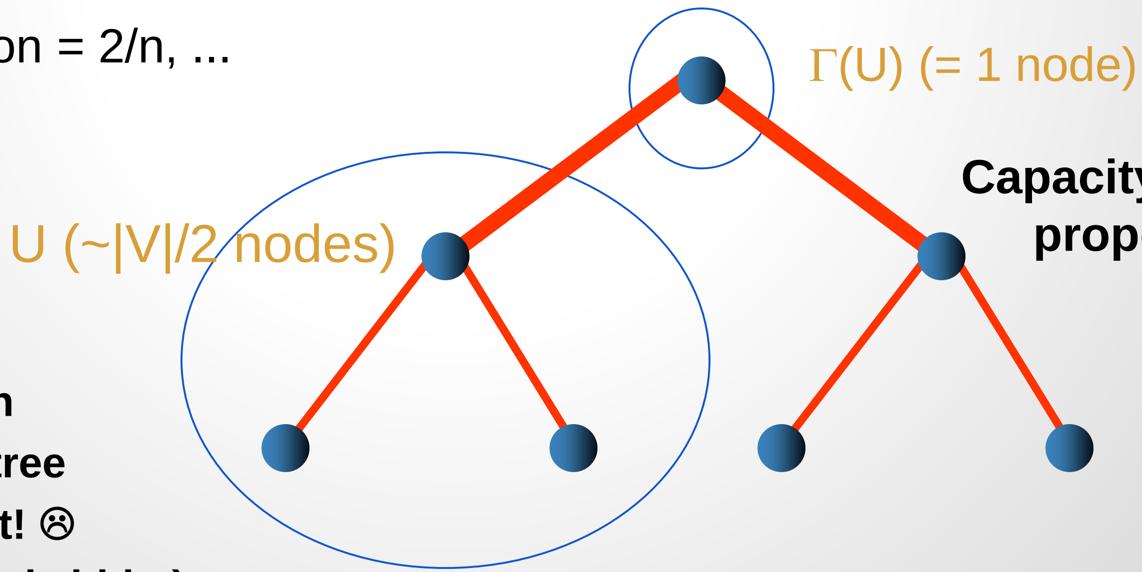


Degree? Diameter? Expansion?

Pro: easy and fast routing (tree = unique paths!), small degree (3), log diameter...

Cons: bad expansion = 2/n, ...

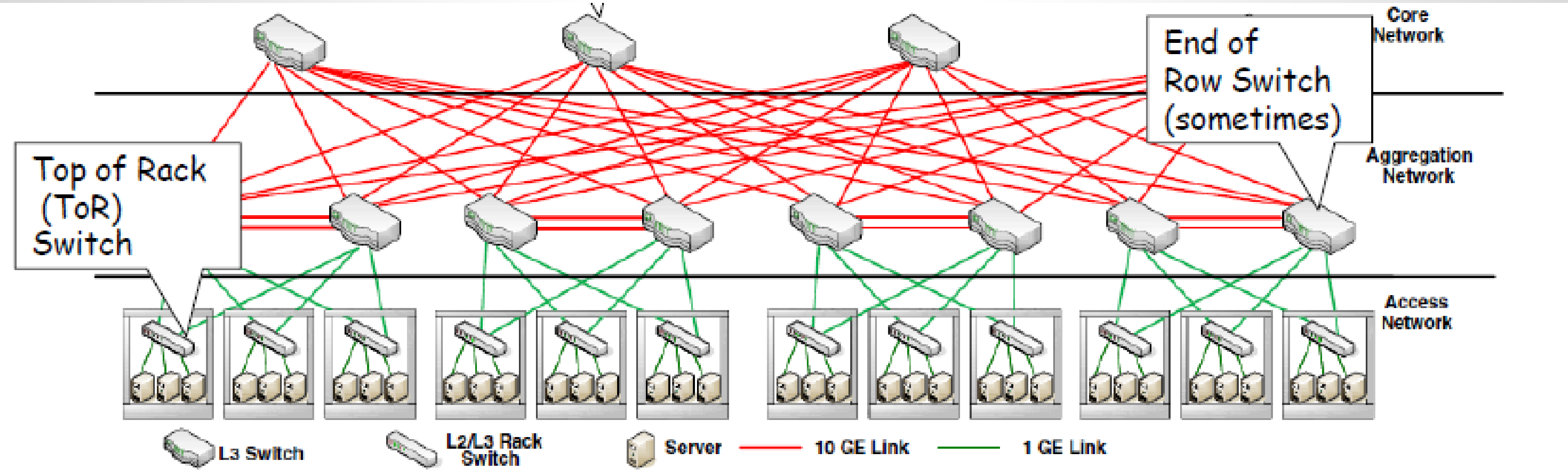
Expansion:



Capacity increases proportionally!

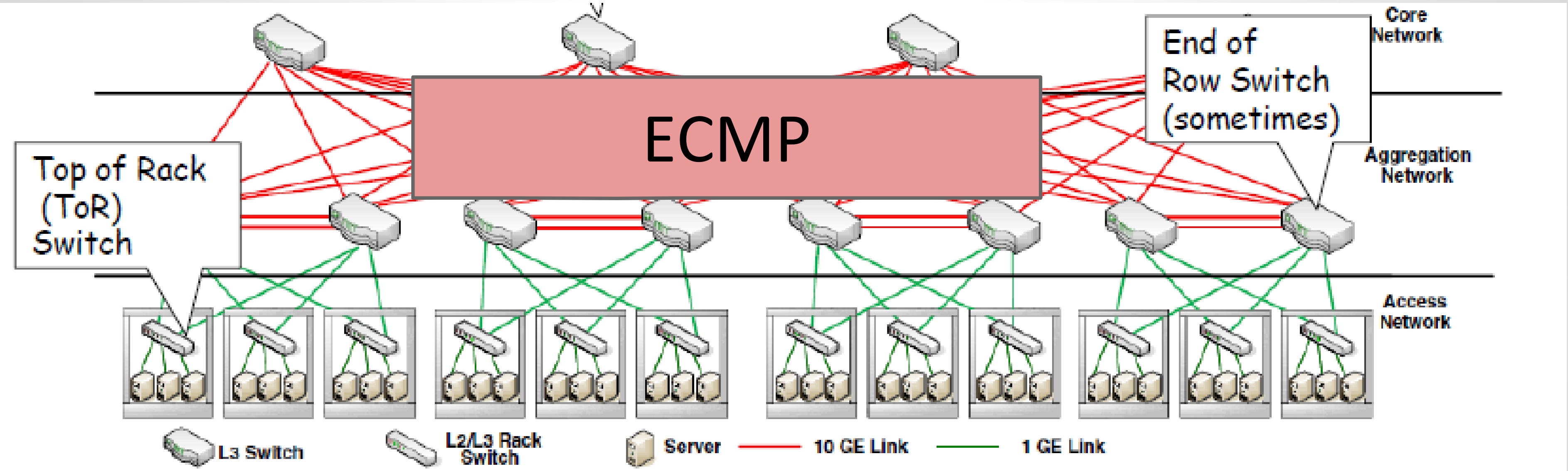
All communication from left to right tree goes through root! ☹ (no «bisection bandwidth»)

Fat-Tree Networks in Reality



Source: K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architectures," Concurrency and Computation: Practice and Experience,

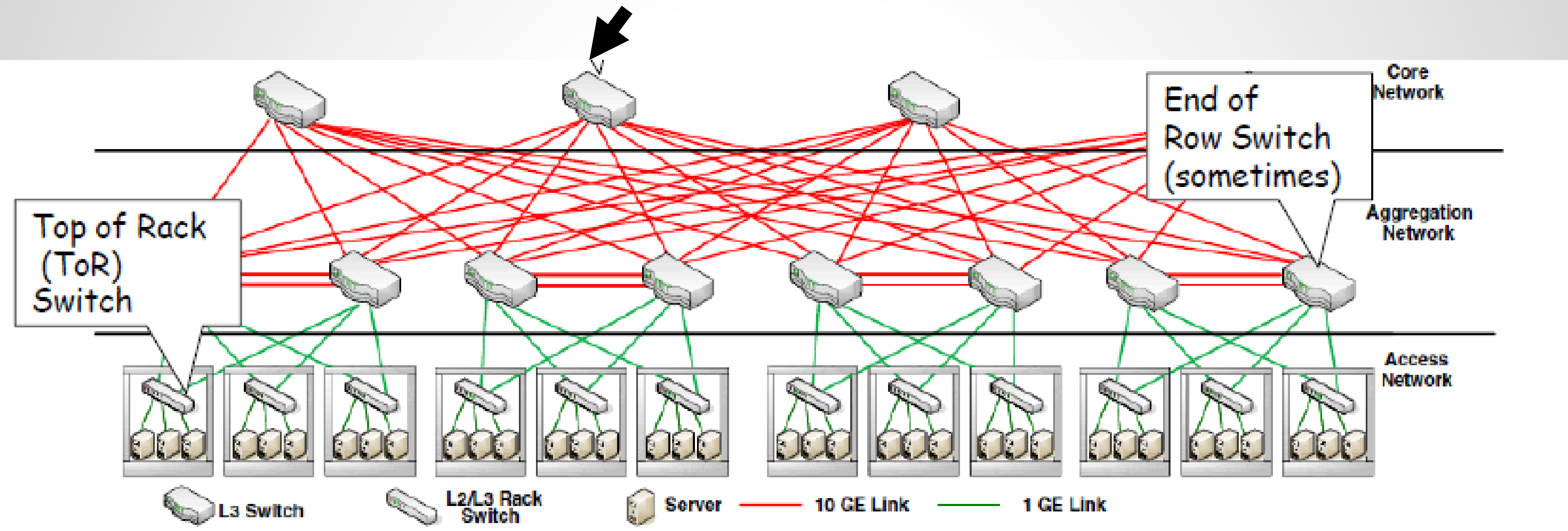
Fat-Tree Networks in Reality



Source: K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architectures," Concurrency and Computation: Practice and Experience,

Fat-Tree Networks in Reality

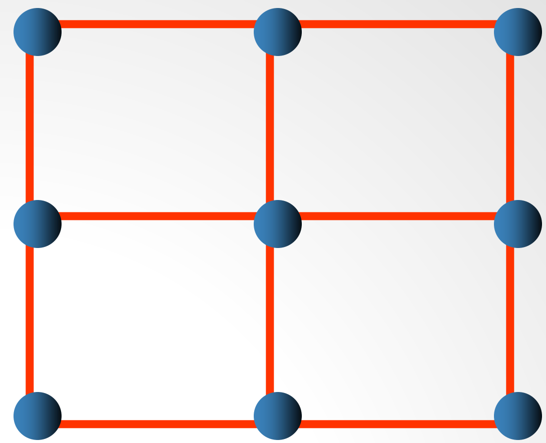
For some extra EUR,
these can be routers.



Source: K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architectures," Concurrency and Computation: Practice and Experience,

More Examples…

2d Mesh: pro and cons?



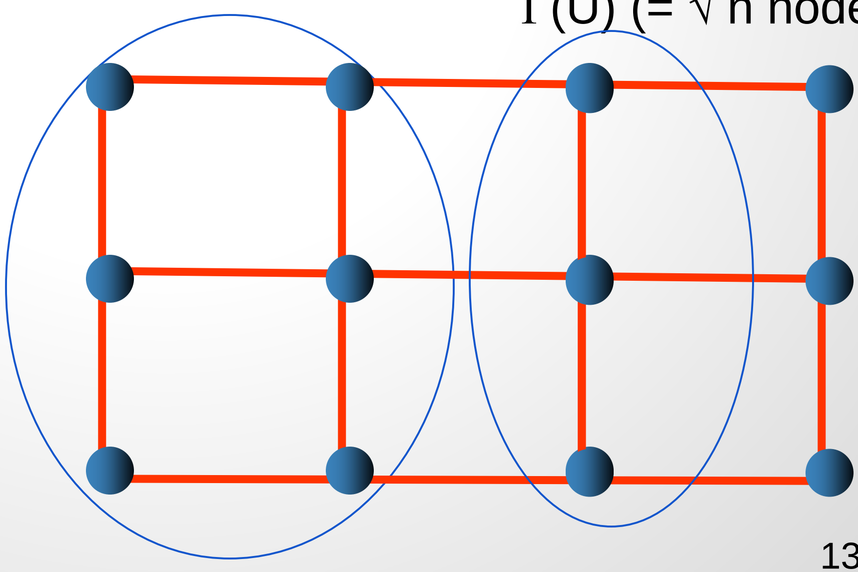
Degree? Diameter? Expansion?

Pro: easy and fast routing (coordinates!), small degree (4), <2 $\sqrt{n}$ diameter...
Cons: diameter?, expansion = ~$2/\sqrt{n}$, ...

$\Gamma(U)$ (= $\sqrt{n}$ nodes)

Expansion:

U (~n/2 nodes)



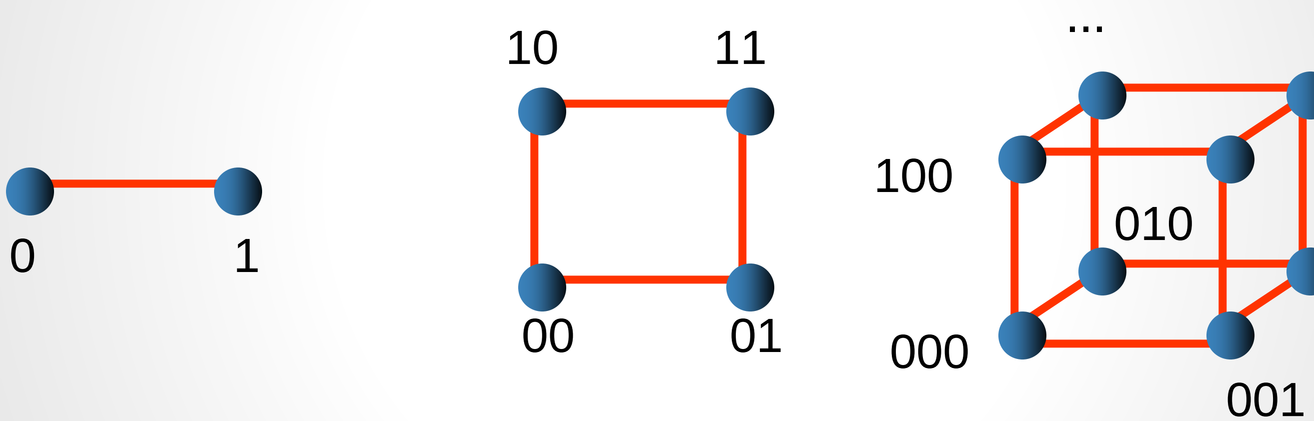
Future Datacenters: Hypercubic

d-dim Hypercube: Formalization?

Nodes $V = \{(b_1,...,b_d), b_i \text{ binary}\}$ (nodes are bitstrings!)

Edges E = for all i: $(b_1,..., b_i, ..., b_d)$

connected to $(b_1, ..., 1\text{-}b_i, ..., b_d)$



Degree? Diameter? Expansion? How to get from (100101) to (011110)?

2^d = n nodes => d = log(n): degree

Diameter: fix one bit after another => log(n) too

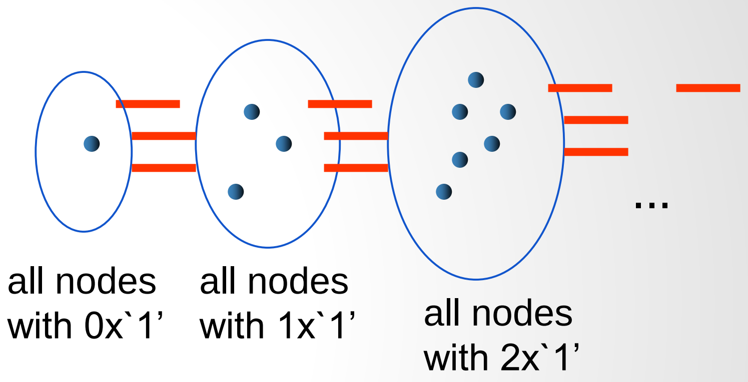
Hypercube: Expansion (upper bound for a ball)

d-dim Hypercube:

Nodes V = {(b_d,...,b_1), b_i 2 {0,1}}

Edges E = for all i: (b_d,..., b_i, ..., b_1)

connected to (b_d, ..., 1-b_i, ..., b_1)



Expansion? Find small neighborhood!

1/ √ d=1/ √ log n

Idea: nodes with i x`1’ are connected to which nodes?

To nodes with (i-1) x`1’ and (i+1) x`1’...:

Hypercube: Expansion (upper bound for a ball)

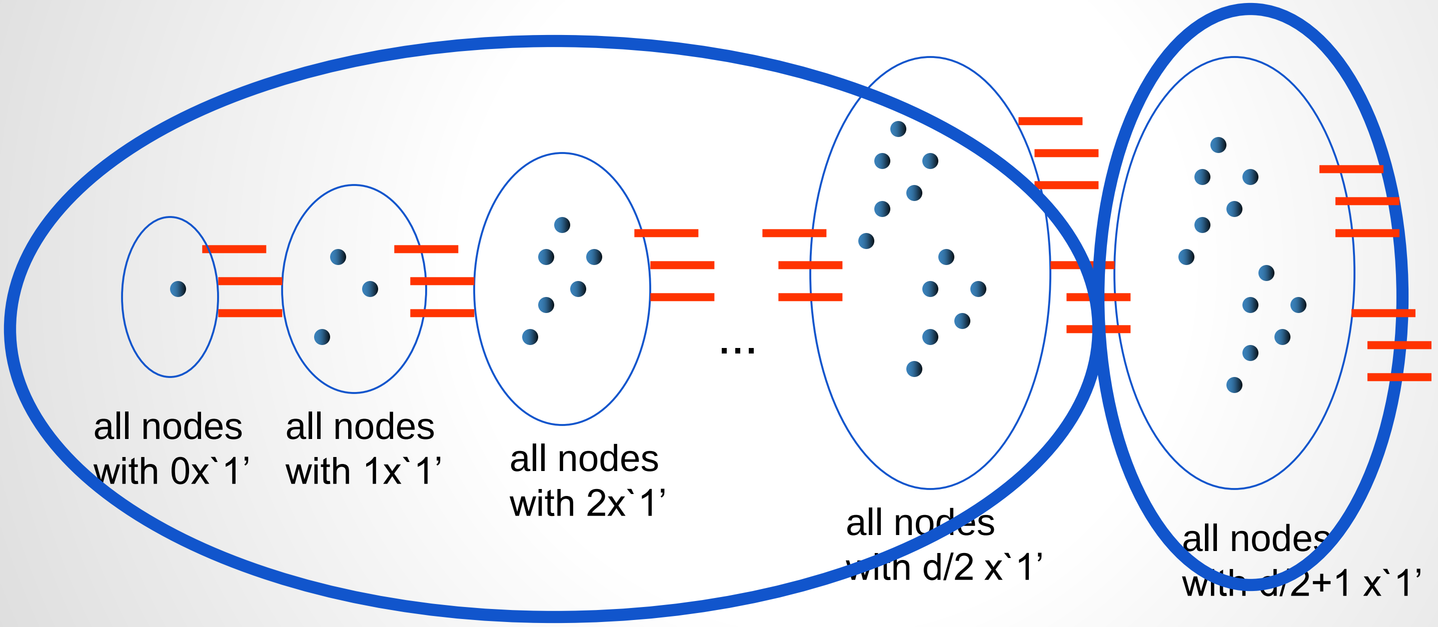
Idea:

How many nodes?

Γ(U) (= ?)

U (~n/2 nodes)

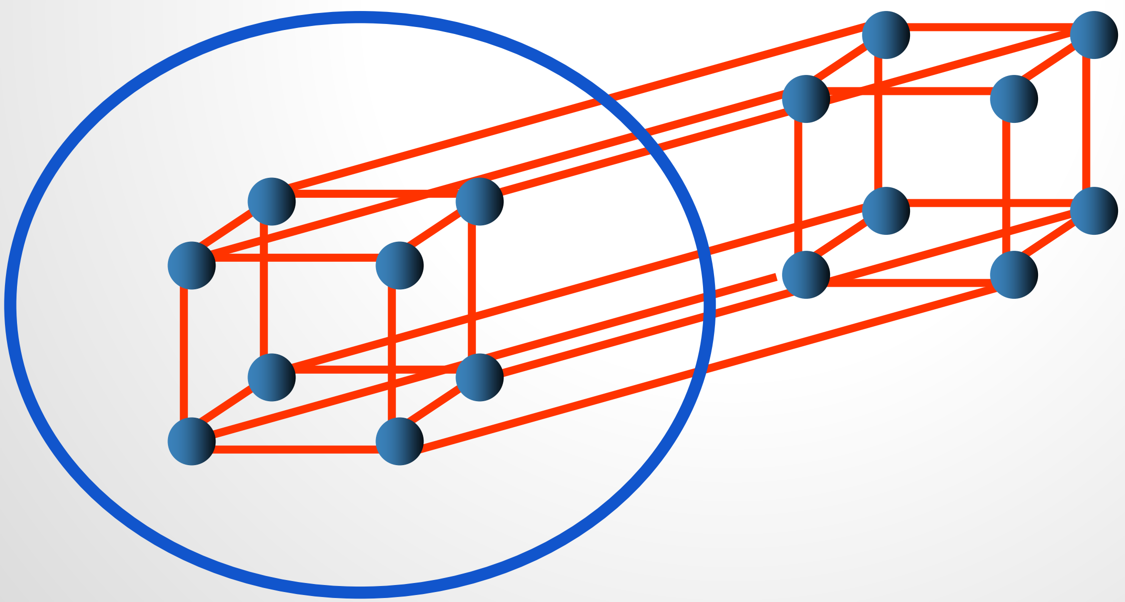
= binomial(d,d/2+1)



Expansion then follows from computing the ratio...

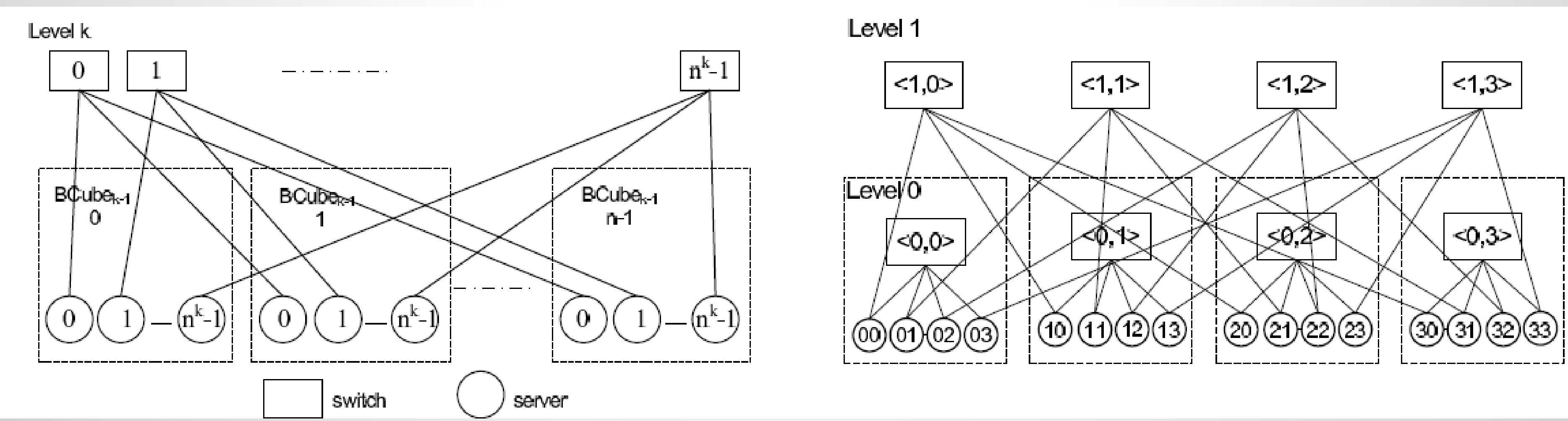
Hypercube: Expansion

- The expansion $1/\sqrt{\log n}$ is an upper bound only
- In general, it is just $1/\log n$: In the dimension cut, 1 out of $\log n$ edges crosses the dimension



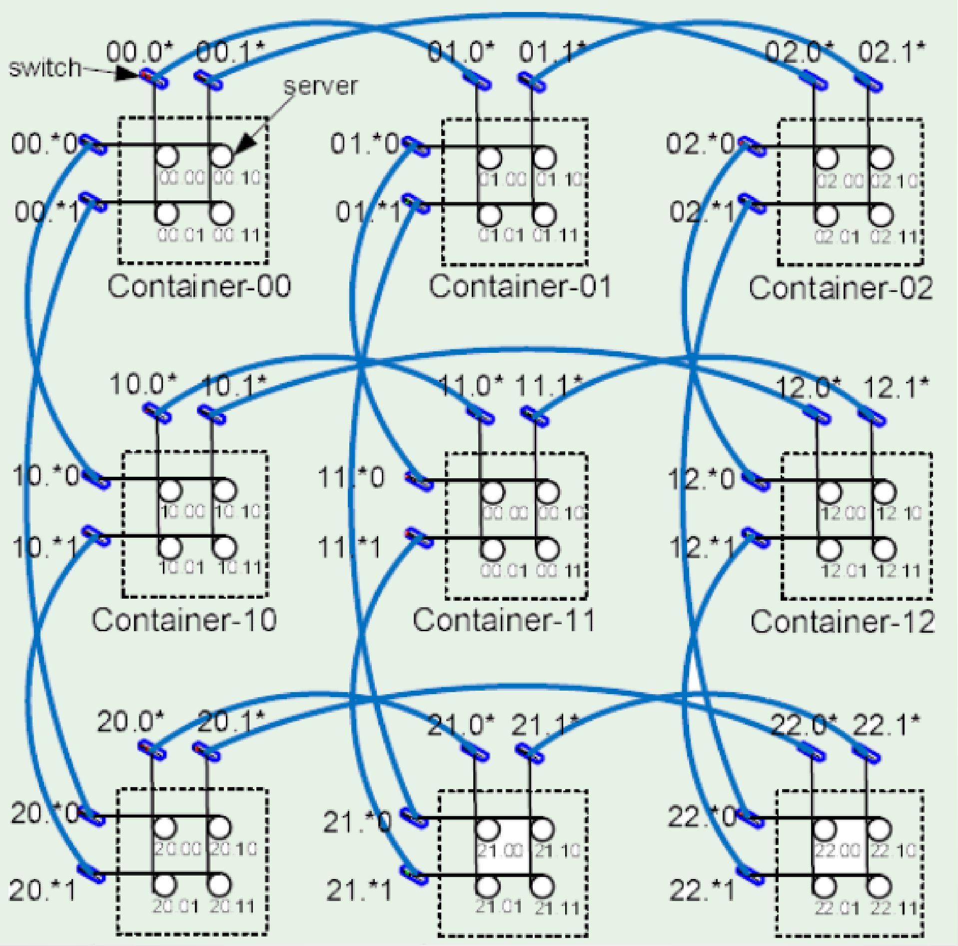
Datacenter Topologies

- Many more recently proposed datacenter topologies are hypercubic: BCube, MDCube, even Jellyfish
- Example: BCube
  - Modular design: based on shipping containers
  - Server centric: switches only connect to servers, but not other switches
  - Low-cost, mini-switches



Datacenter Topologies

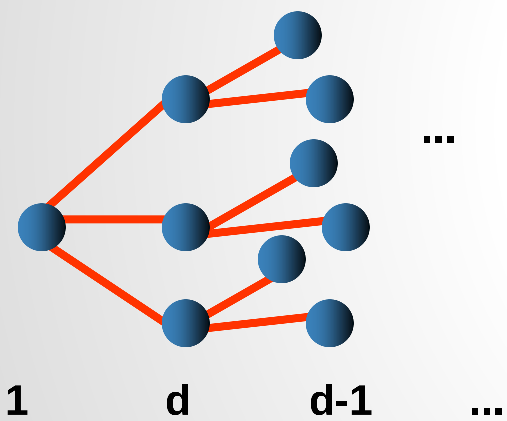
- Example: MDCube



What is the degree-diameter tradeoff? Idea? Proof?

Theorem

Each network with n nodes and max degree d>2 must have a diameter of at least log(n)/log(d-1)-1.



In two steps, at most

d (d-1)

additional nodes can be reached!

So in *k* steps at most:

$$1+\sum_{i=0}^{k-1} d\cdot(d-1)^i = 1+d\cdot\frac{(d-1)^k-1}{(d-1)-1} \leq \frac{d\cdot(d-1)^k}{d-2}$$

To ensure it is connected this must be at least n, so:

$$(d-1)^k \geq \frac{(d-2)\cdot n}{d} \quad \Leftrightarrow \quad k \geq \log_{d-1}\left(\frac{(d-2)\cdot n}{d}\right) \quad \Leftrightarrow \quad k \geq \log_{d-1} n + \log_{d-1}\left(\frac{d-2}{d}\right)$$

Reformulating this yields the claim... ☺

What is the best tradeoff? E.g., Pancake Graphs

Graph which minimizes max(degree, diameter)!

Both in O(log n / log log n)

Nodes = permutations of {1,...,d}

Edges = prefix reversals

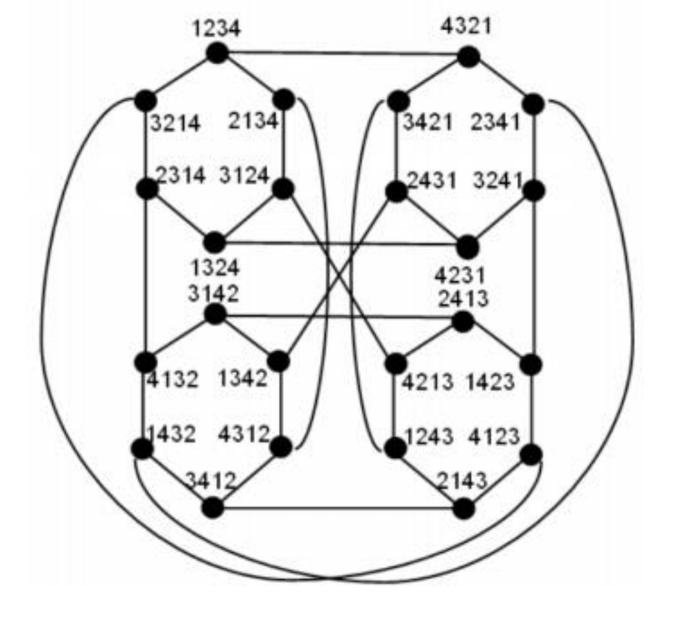
nodes? degree?

d! many nodes and degree (d-1).

Routing?

E.g., from (3412) to (1243)?

Fix bits at the back, one after the other, in two steps, so diameter also log n / log log n.



d! = n,
so by Stirling formula:
d = log(n)/loglog(n)
(insert it to d^d=n, resp. to
d log(d) = log(n)...)

Drawing Pancake Graphs

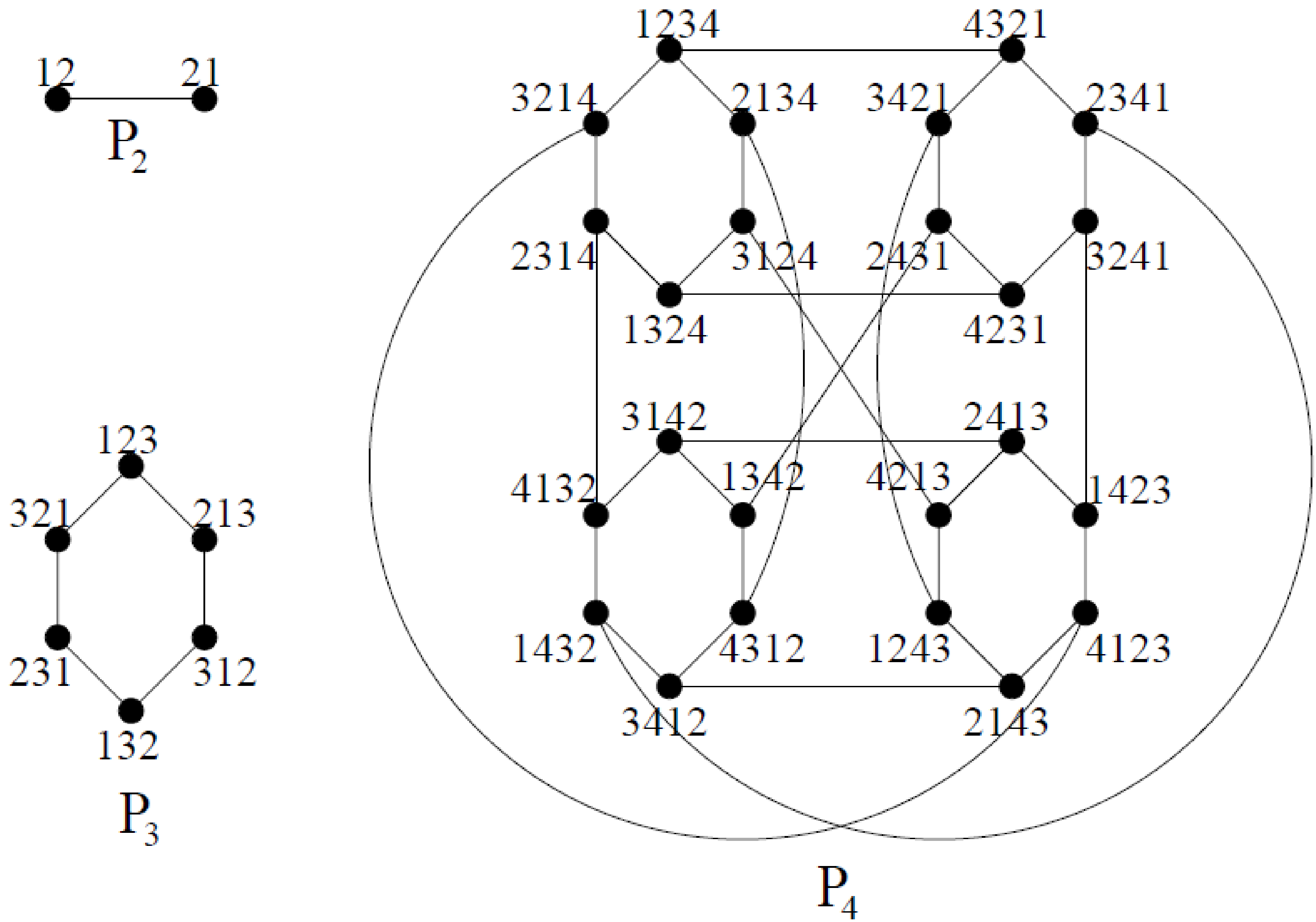
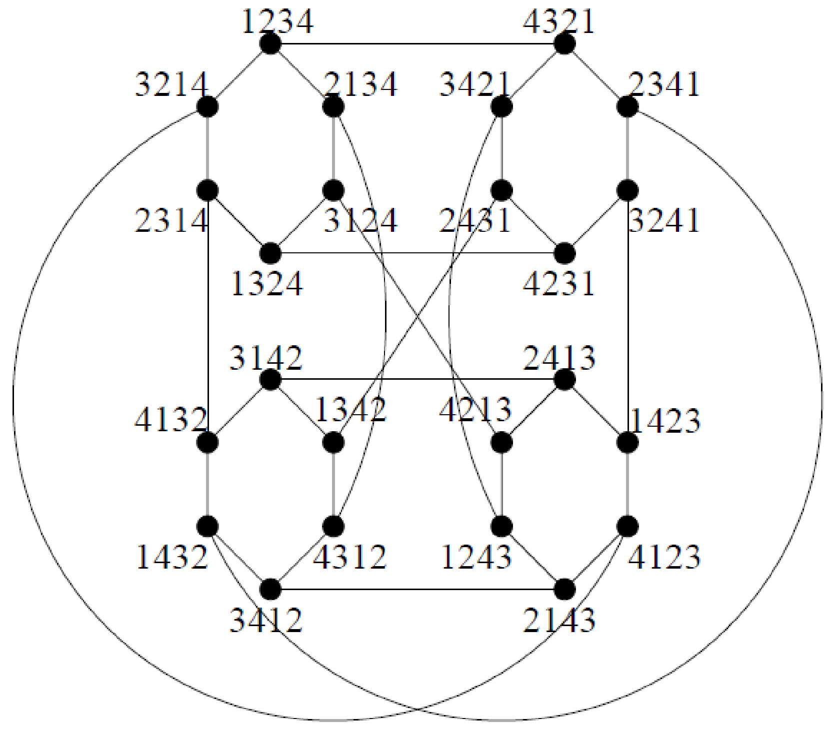


Figure 1: Pancake graphs for $n = 2, 3, 4$.

Degree Pancake Graphs

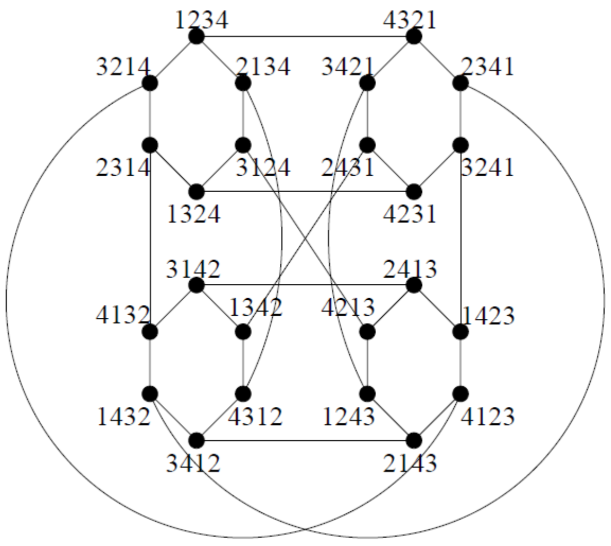
There are n-1 non-trivial prefix reversals for an n-dimensional Pancake with n digits



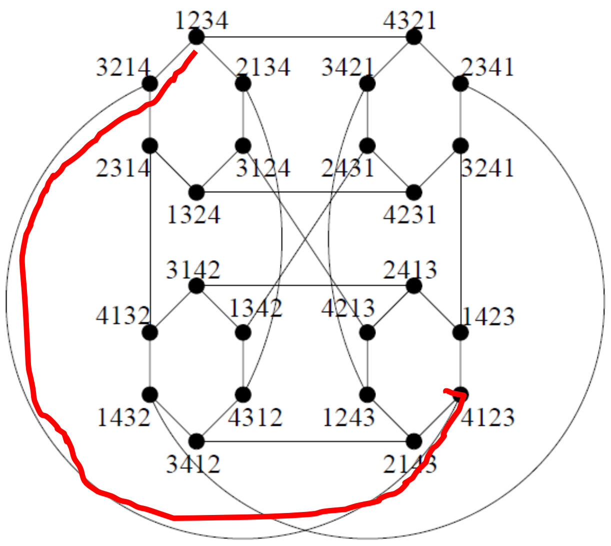
P_4

Diameter of Pancake Graphs

- How to get from node $v=v_1...v_n$ to $w=w_1...w_n$?
- Idea: fix one digit after the other at the back!
  - Two steps: Prefix reversal such that digit is at the front, then full prefix reversal such that digit is at the back
- Length of routing path: at most 2*(n-1)
- Papadimitriou and Bill Gates have shown that this is asymptotically also optimal (i.e., close to diameter)



P_4



P_4

Rough Plan

- ❑ SDN and Network Virtualization: Debunking some myths and misunderstandings
- ❑ Some fundamental research challenges
- ❑ **Mini-Tutorial: How to exploit flexiblities in virtualized networked systems? ☺**
- ❑ Mini-Tutorial: How are datacenters designed?
- ❑ Mini-Tutorial: Put your hands on SDN

How to Embed a VNet in a Typical Datacenter Topology?

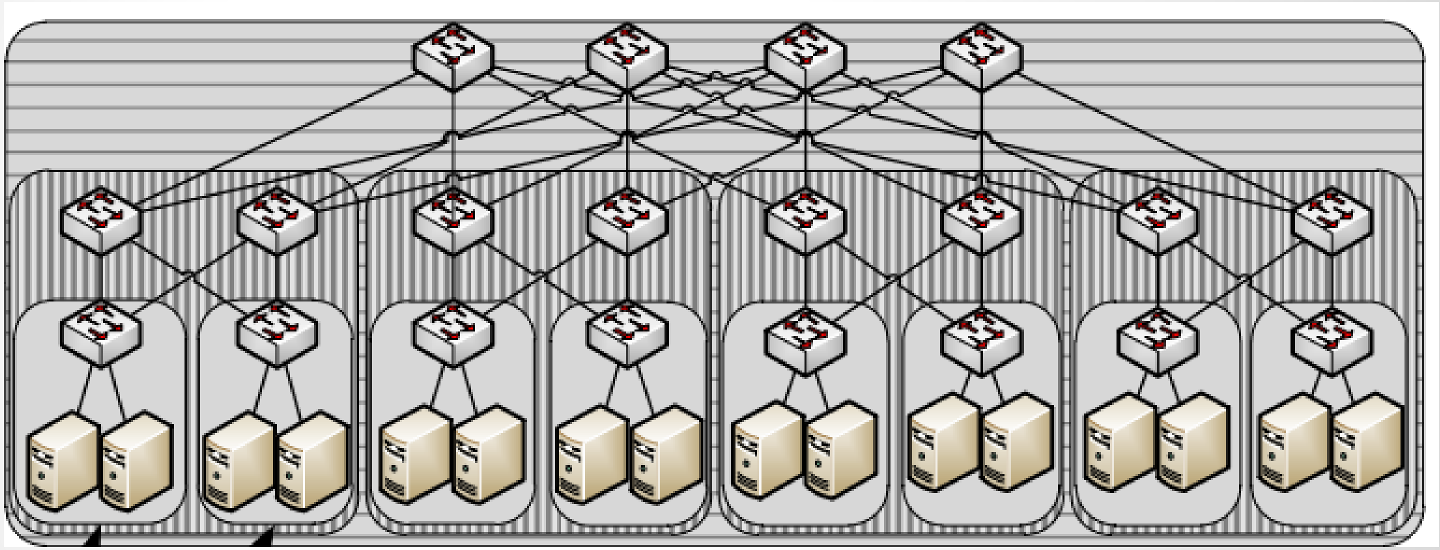
- If realized with multiple commodity switches and links

core routers

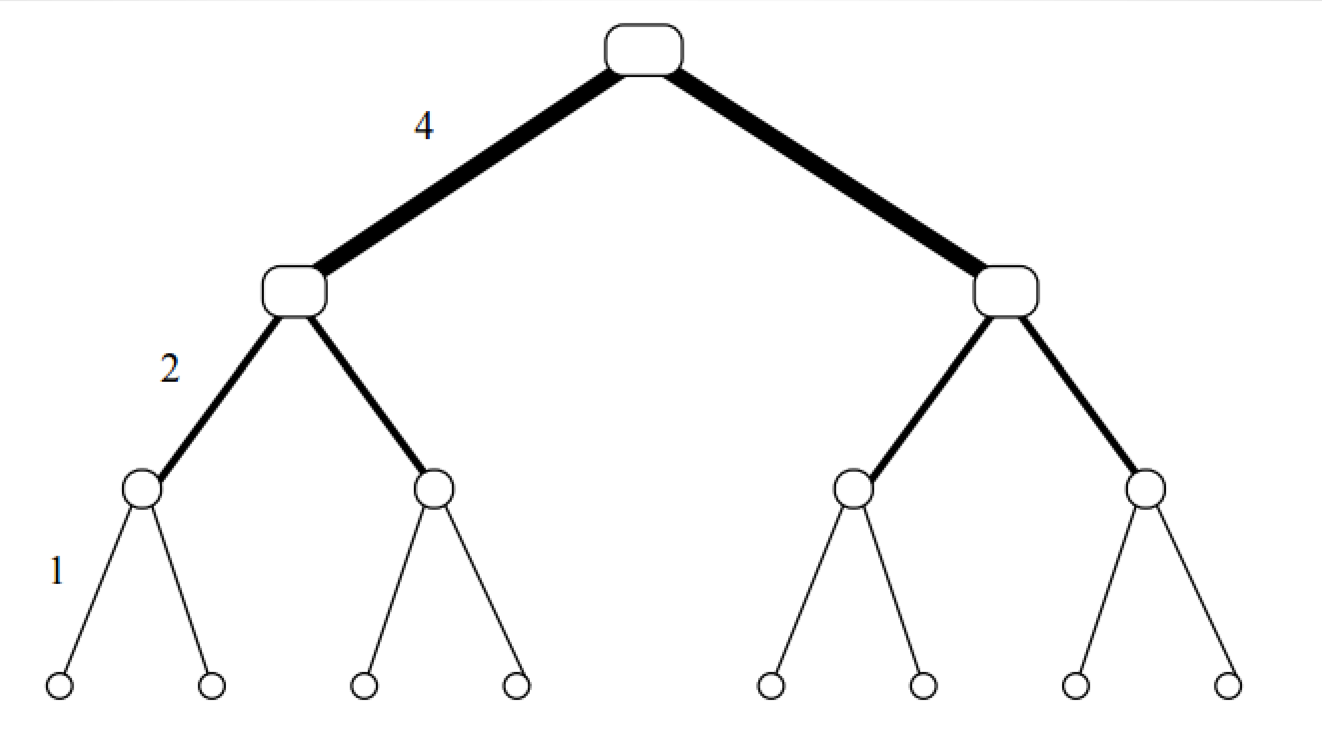
pods

top-of-rack

Servers at edge



A Typical Datacenter Topology

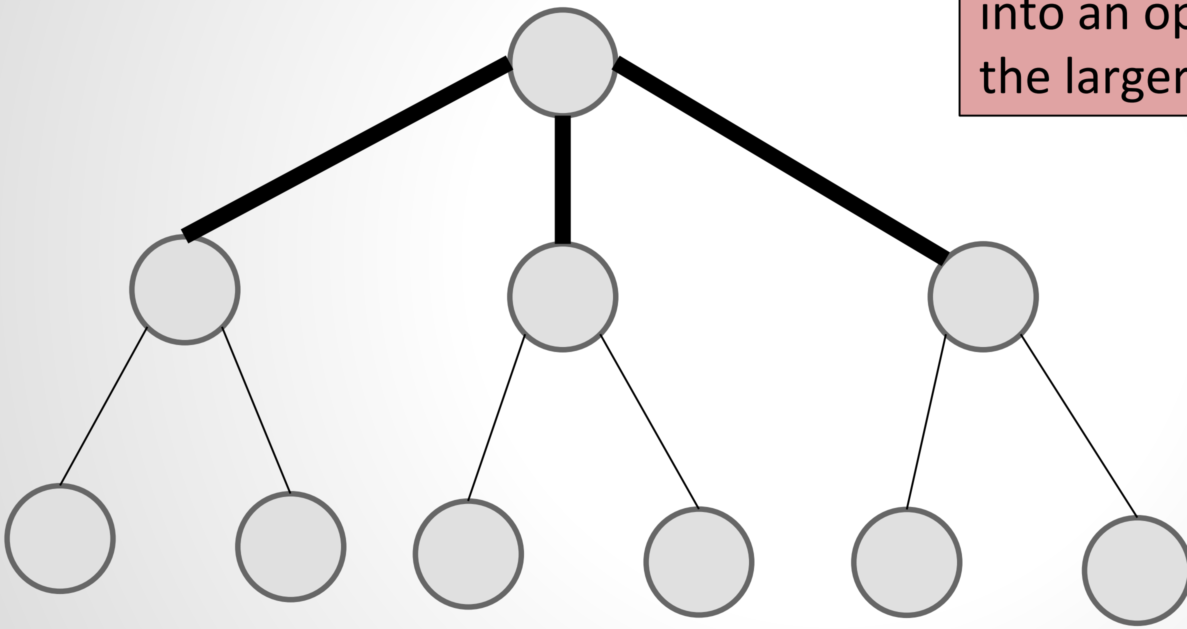


But due to ECMP, often ok to think of it like this.

How to embed a Virtual Cluster in a Fat-Tree?

- Example: dynamic programming

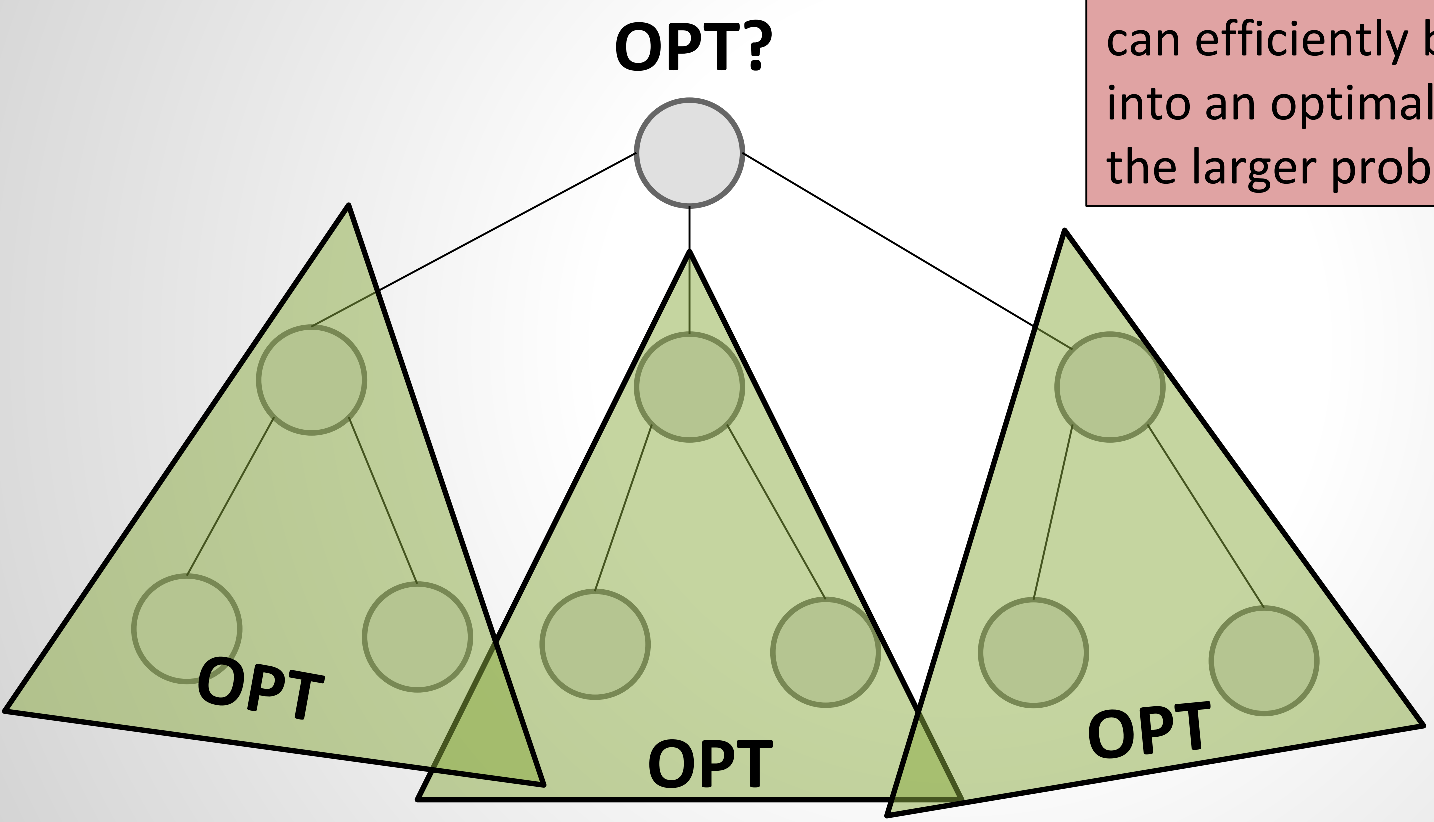
Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!



How to embed a Virtual Cluster in a Fat-Tree?

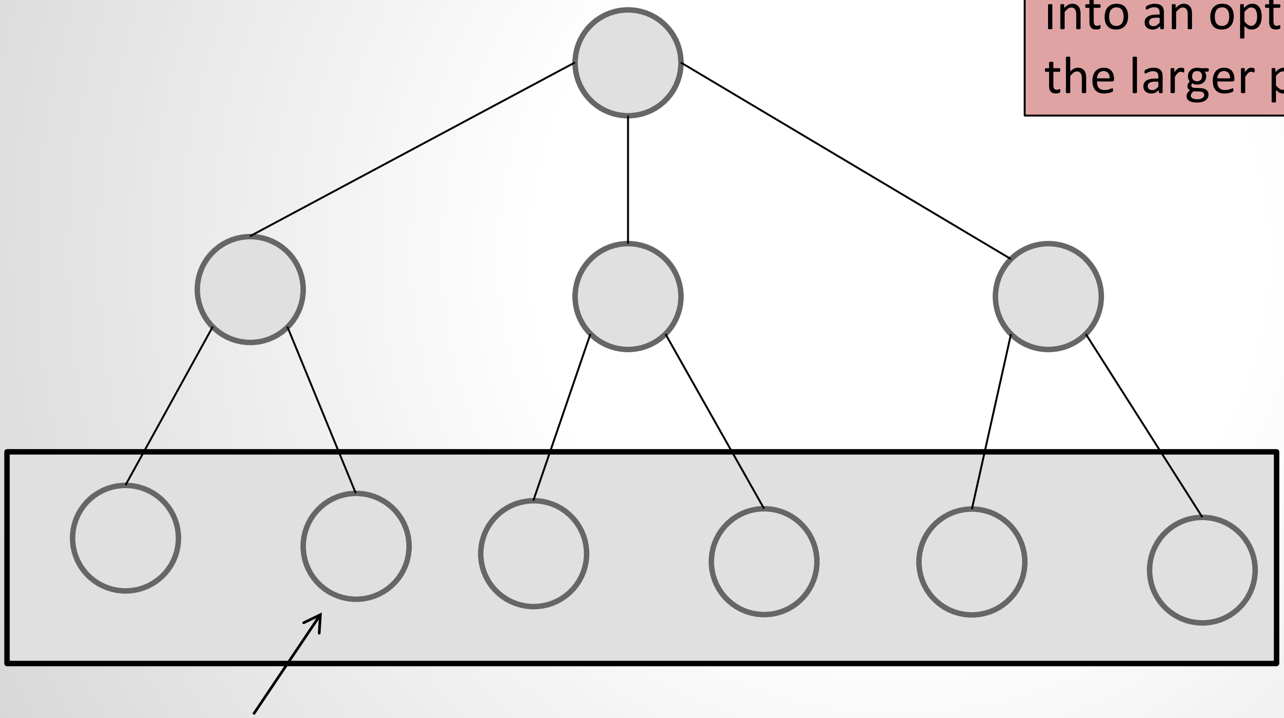
- Example: dynamic programming

Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!



How to embed a Virtual Cluster in a Fat-Tree?

Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!



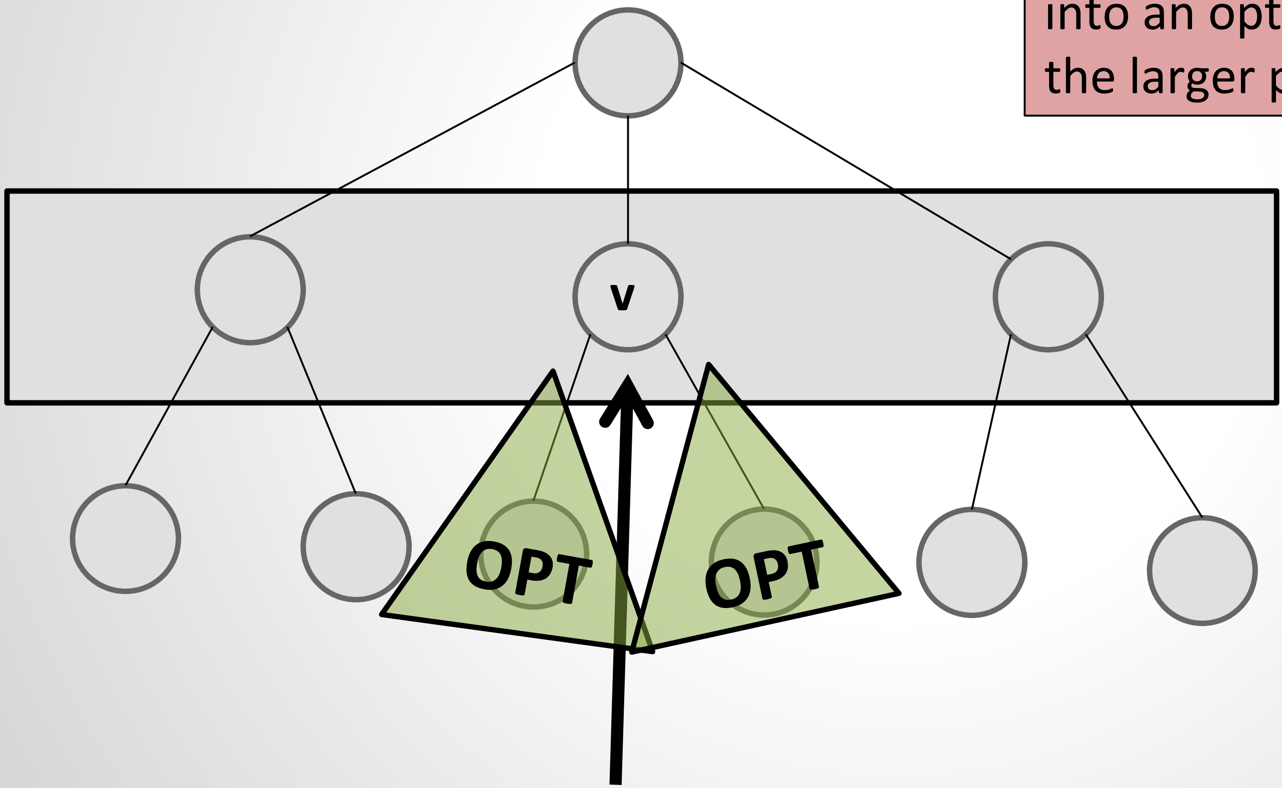
t = 0: solve leaves!

How to optimally embed x VMs here, $x \in \{0, ..., n\}$?

Cost = 0 or ∞!

How to embed a Virtual Cluster in a Fat-Tree?

Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!



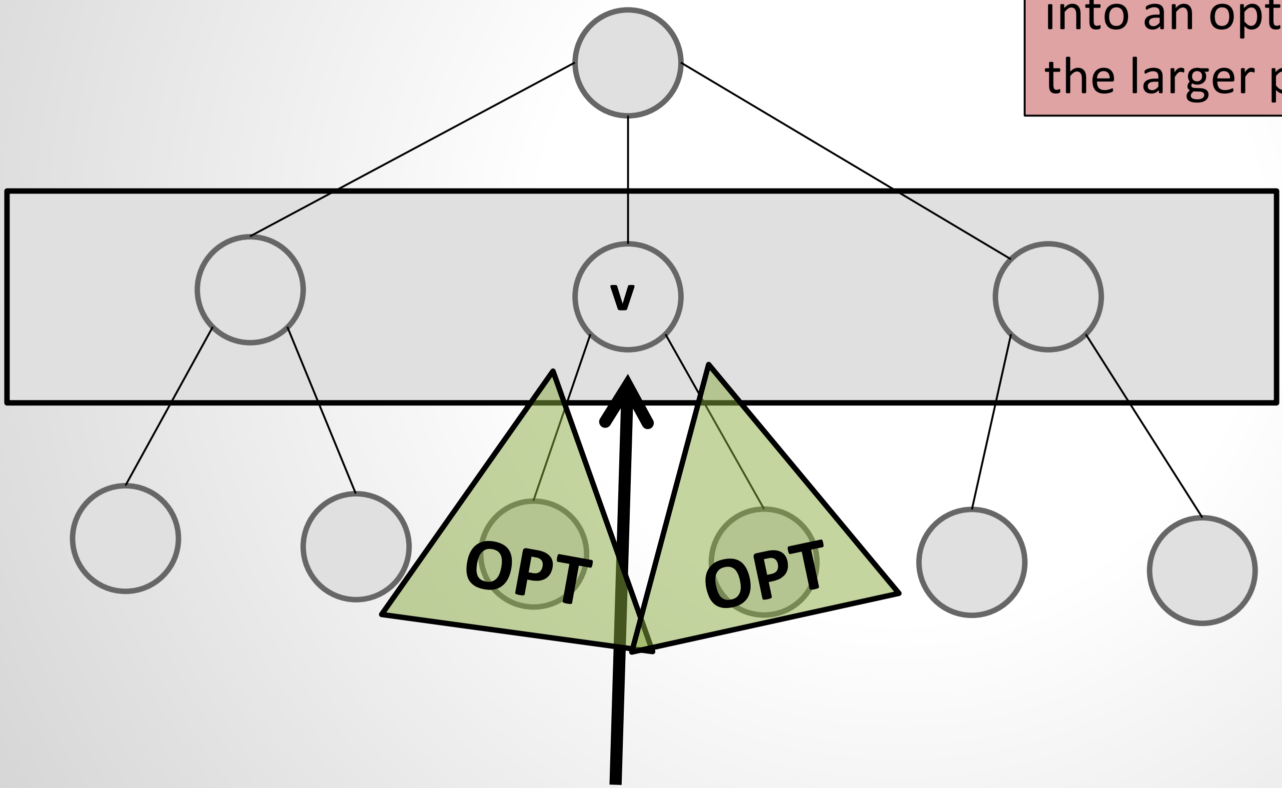
t = 1: solve height 1!

Cost[x] = $\min_y$ Cost[y] + Cost[x-y]

+ cross-traffic + connections to v

How to embed a Virtual Cluster in a Fat-Tree?

Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!



t = 1: solve height 1!

$\text{Cost}[x] = \min_y \text{Cost}[y] + \text{Cost}[x\text{-}y]$

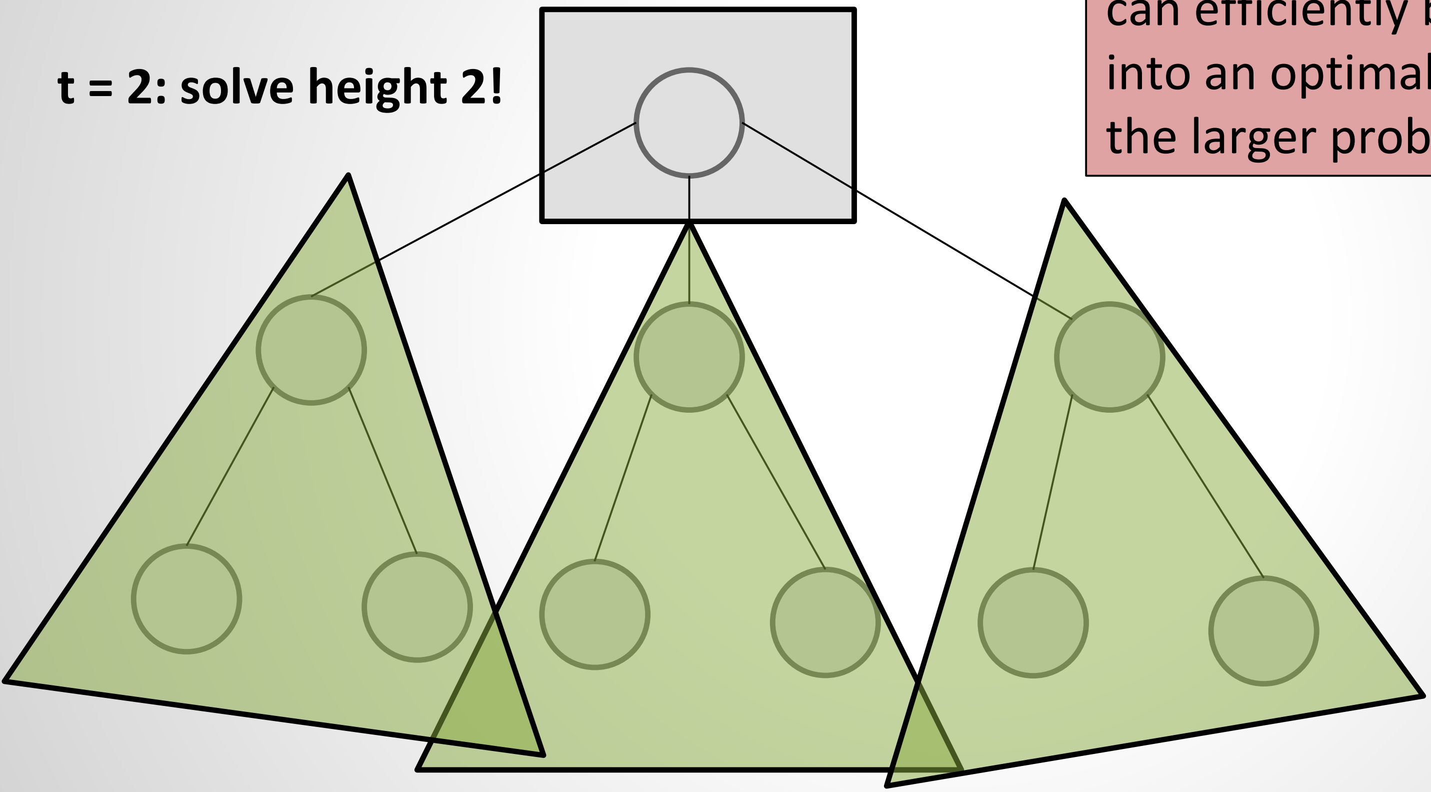
$+$ cross-traffic + connections to v

Or just account on upward link (number of leaving links!)

How to embed a Virtual Cluster in a Fat-Tree?

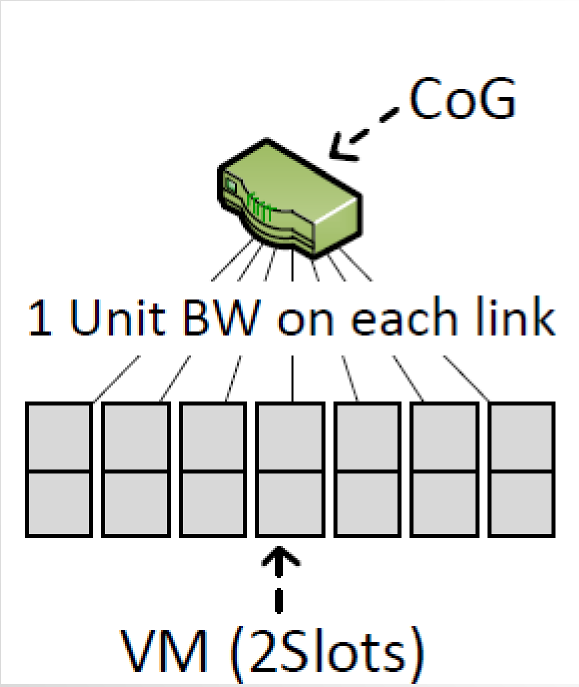
Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!

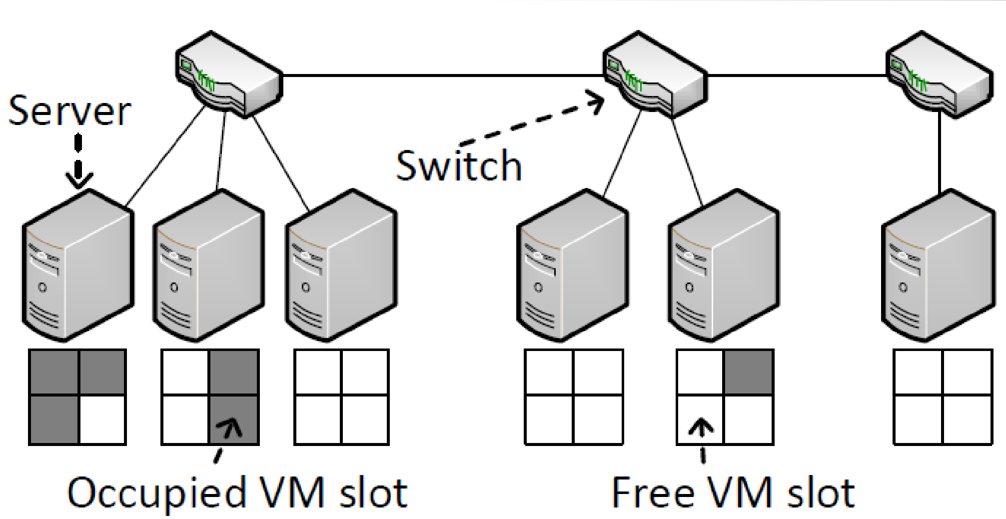
t = 2: solve height 2!



How to embed a Virtual Cluster in a General Graph?

How to embed?





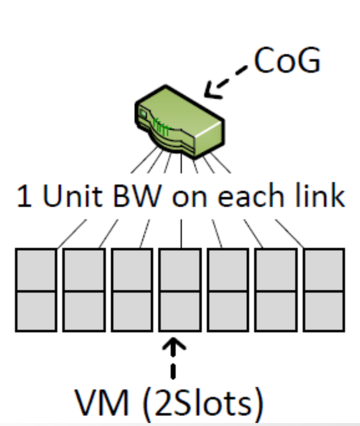
Host Graph

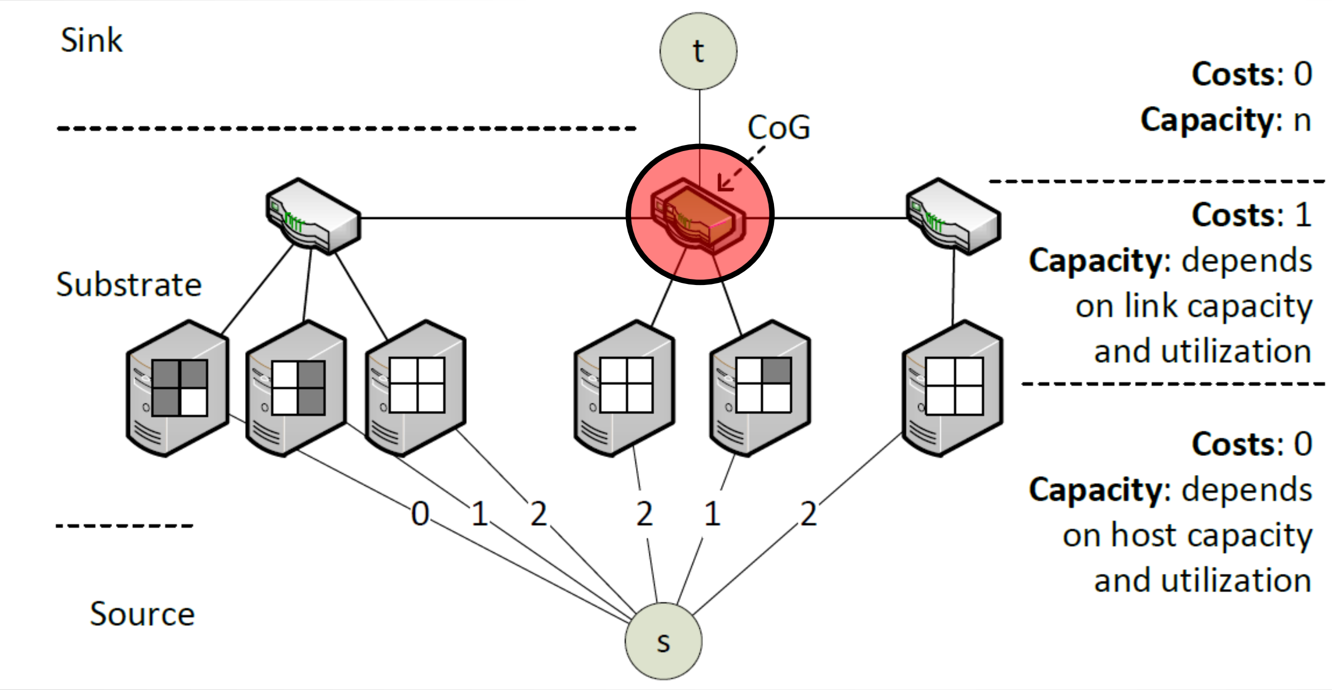
Guest Graph

How to embed a Virtual Cluster in a General Graph?

Algorithm:

- Try all possible locations for virtual switch
- Extend network with artificial source s and sink t
- Add capacities
- Compute min-cost max-flow from s to t
  (or simply: min-cost flow of volume n)



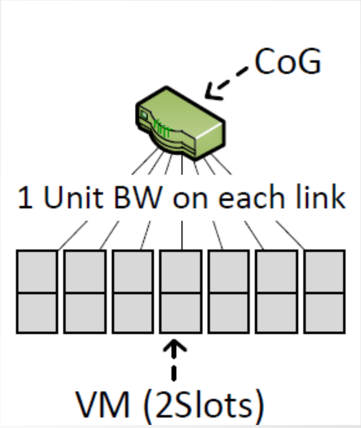


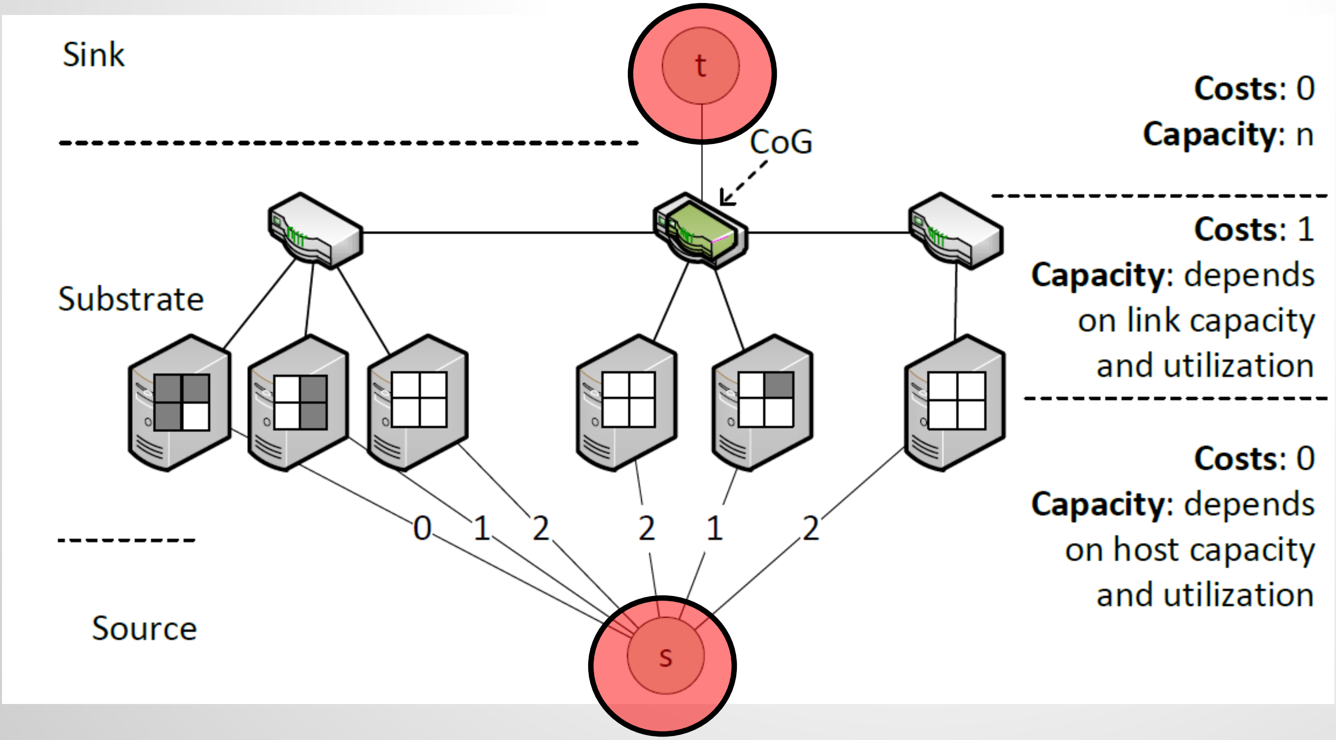
How to embed a Virtual Cluster in a General Graph?

Algorithm:

- Try all possible locations for virtual switch
- Extend network with artificial source s and sink t
- Add capacities
- Compute min-cost max-flow from s to t

  (or simply: min-cost flow of volume n)

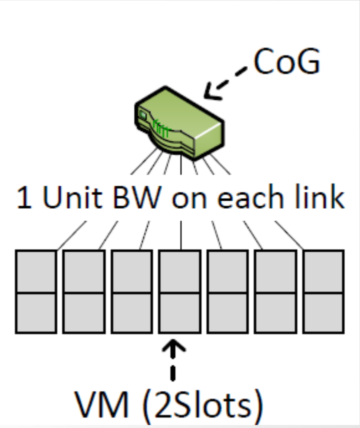


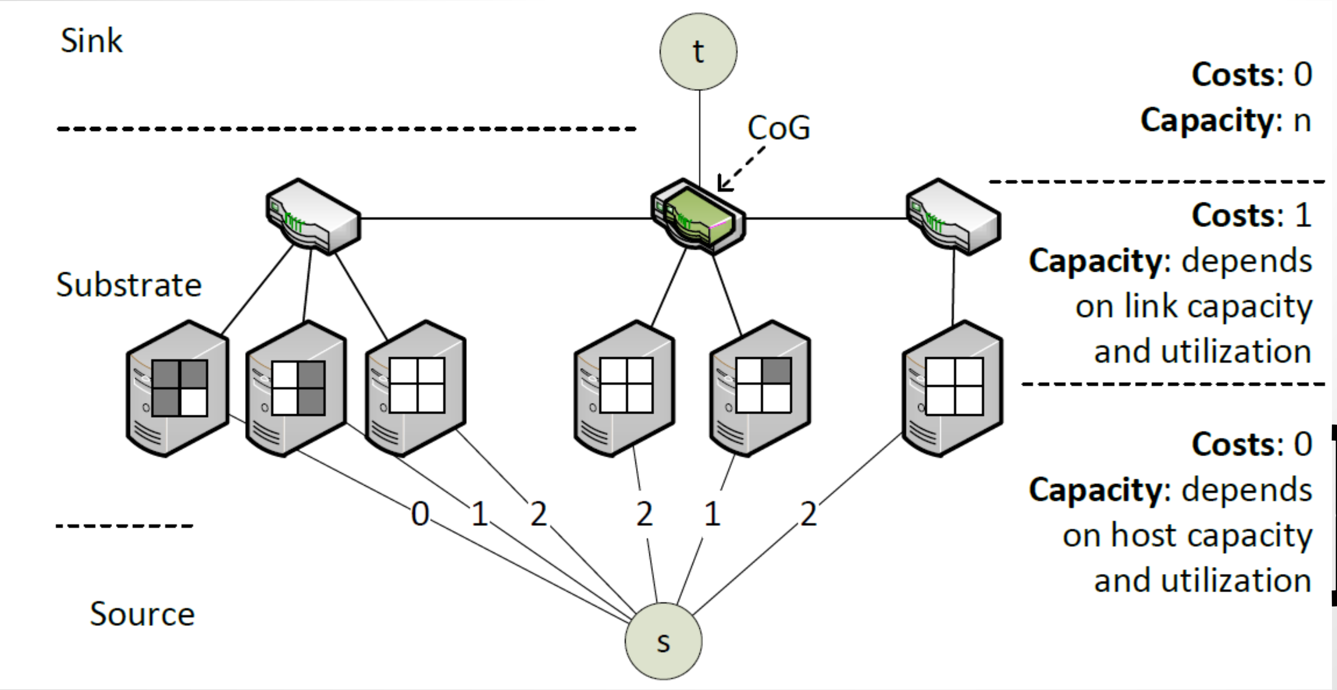


How to embed a Virtual Cluster in a General Graph?

Algorithm:

- Try all possible locations for virtual switch
- Extend network with artificial source s and sink t
- Add capacities
- Compute min-cost max-flow from s to t
  (or simply: min-cost flow of volume n)

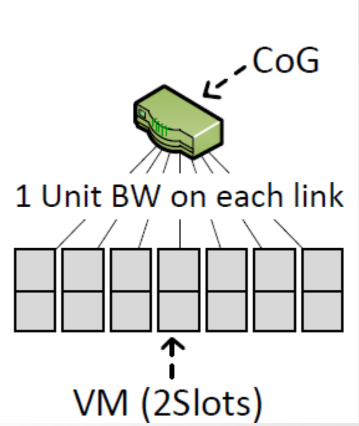




enough to embed n VMs

capacity = floor(available resources / unit demand)

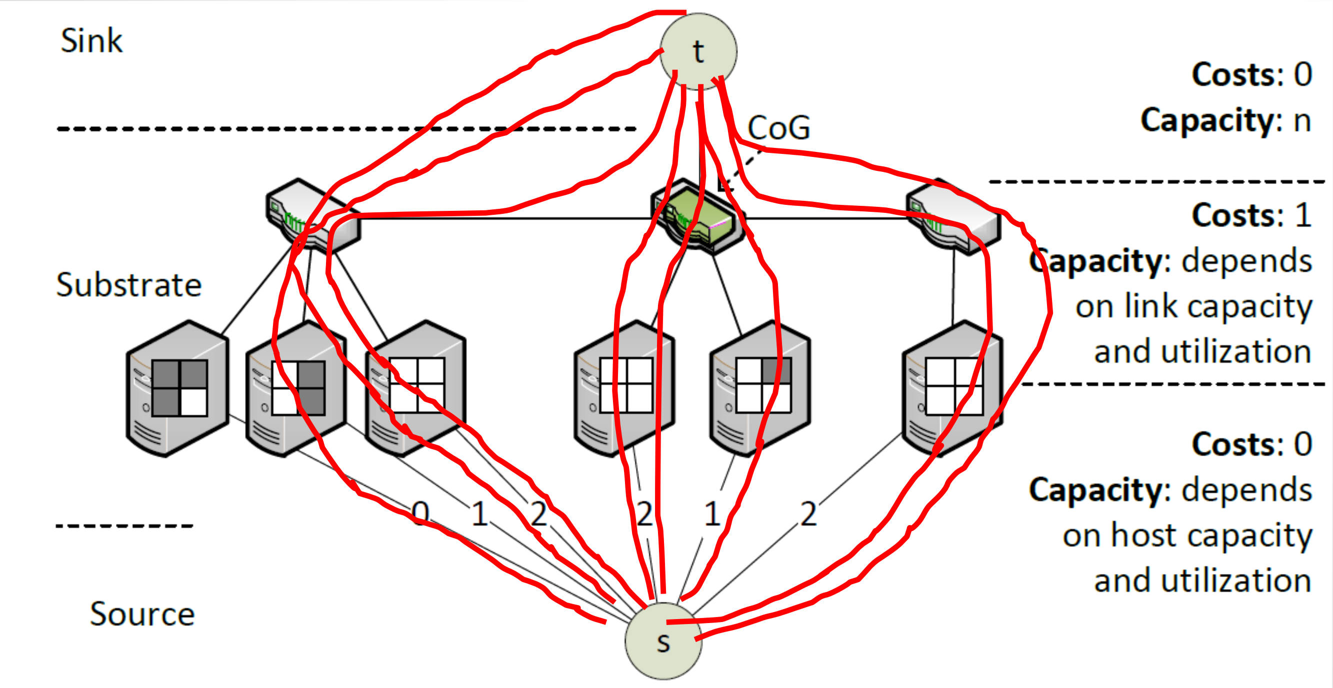
How to embed a Virtual Cluster in a General Graph?



Algorithm:

- Try all possible locations for virtual switch
- Extend network with artificial source s and sink t
- Add capacities
- Compute min-cost max-flow from s to t

  (or simply: min-cost flow of volume n)



Guaranteed integer if links are integer! (E.g., successive shortest paths)

Reading / Literature Pointer

- Beyond the Stars: Revisiting Virtual Cluster Embeddings
  Matthias Rost, Carlo Fuerst, and Stefan Schmid.
  ACM SIGCOMM Computer Communication Review (**CCR**), July 2015.

- It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities
  Matthias Rost, Stefan Schmid, and Anja Feldmann.
  28th IEEE International Parallel and Distributed Processing Symposium (**IPDPS**), Phoenix, Arizona, USA, May 2014.

- Optimizing Long-Lived CloudNets with Migrations
  Gregor Schaffrath, Stefan Schmid, and Anja Feldmann.
  5th IEEE/ACM International Conference on Utility and Cloud Computing (**UCC**), Chicago, Illinois, USA, November 2012.

- How Hard Can It Be? Understanding the Complexity of Replica Aware Virtual Cluster Embeddings
  Carlo Fuerst, Maciek Pacut, Paolo Costa, and Stefan Schmid.
  23rd IEEE International Conference on Network Protocols (**ICNP**), San Francisco, California, USA, November 2015.

Solving the VNEP

- Formulate a Mixed Integer Program!
- Leverage additional structure!
- **Use online primal-dual approach**

Guarantees Over Time

- How to provide guarantees over time?
- Realm of online algorithms and competitive analysis
  - Input to algorithm: sequence σ (e.g., sequence of requests)
  - Online algorithm ON does not know requests t'>t
  - Needs to be perform close to optimal offline algorithm OFF who knows future!

Competitive Analysis

.

Competitive ratio ρ: max over all possible sequences σ

ρ = Cost(ON)/Cost(OFF)

Guarantees Over Time

- How to provide guarantees over time?

- Realm of online algorithms and competitive analysis

Nice: If competitive ratio is low, there is no need to develop any sophisticated prediction models (which may be wrong anyway)! The guarantee holds in the worst-case.

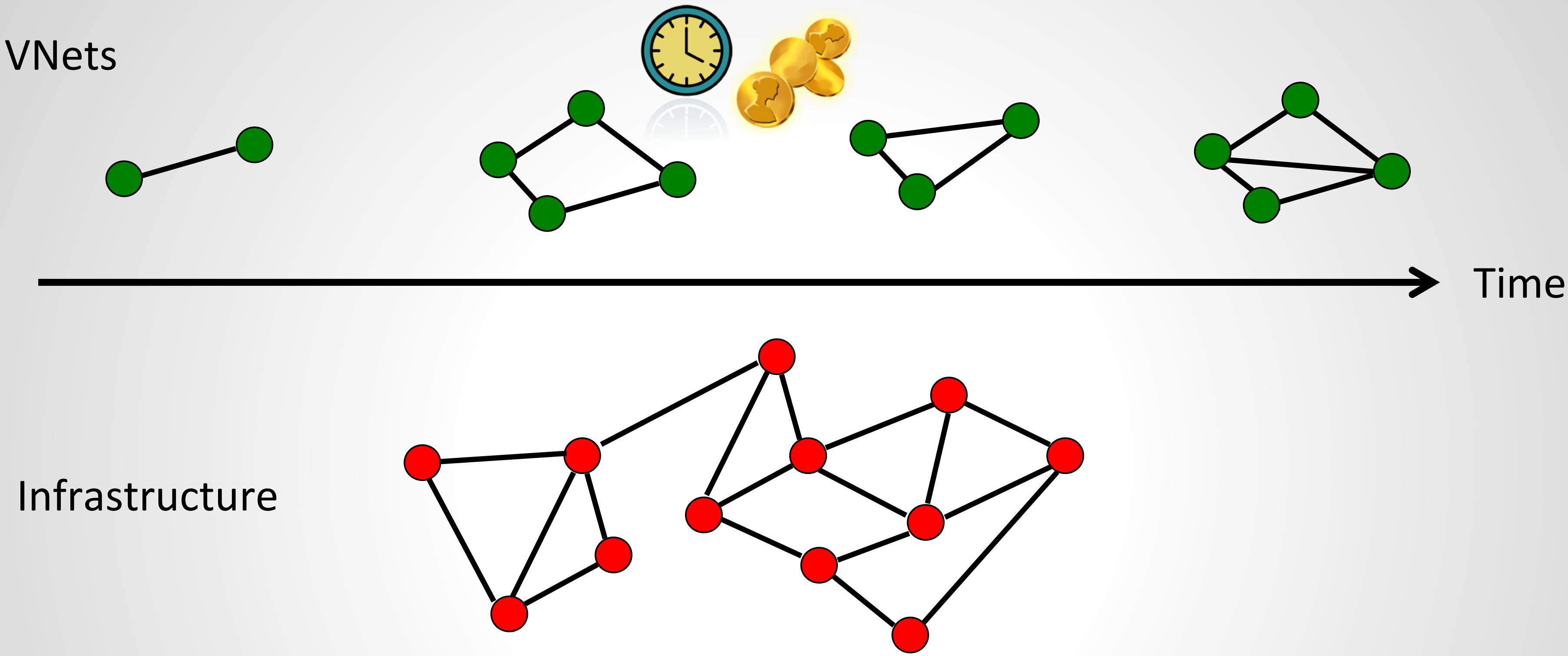
Competitive Analysis

.

Competitive ratio ρ: max over all possible sequences σ

ρ = Cost(ON)/Cost(OFF)

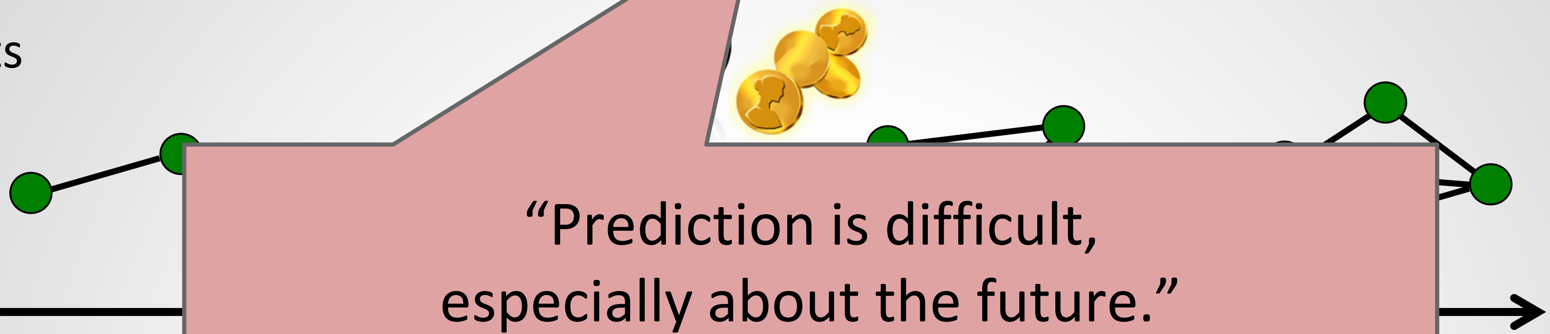
Online Access Control (1)



- Assume: end-point locations given
- Different routing and traffic models
- Price and duration
- Which ones to accept?
- Online Primal-Dual Framework (Buchbinder and Naor)

Online Access Control (1)

VNets



Time

Infrastructur

“Prediction is difficult,
especially about the future.”

Nils Bohr

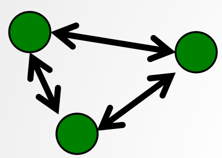
- Assume:
- Differen
- Price and duration
- Which ones to accept?
- Online Primal-Dual Framework (Buchbinder and Naor)

Online Access Control (2)

- Traffic models

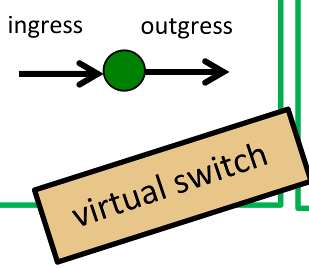
Customer Pipe

Traffic matrix: Bandwidth per VM pair (u,v)



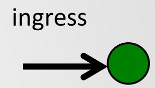
Hose Model

Per VM bandwidth: polytope of traffic matrices.



Aggregate Ingress

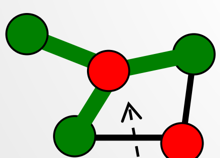
Only ingress specified: e.g., support multicast etc.



- Routing models

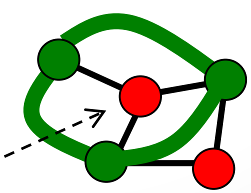
Tree

Steiner tree embedding



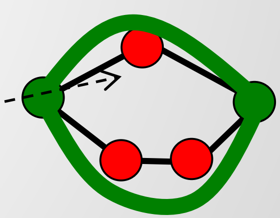
Single Path

Unsplittable paths



Multi-Path

Splittable paths (more capacity)



Relay costs: e.g., depending on packet rate

Online Access Control (3)

Primal and Dual

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Primal and Dual

Online Access Control (3)

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\boxed{\max B_j^T \cdot Y_j \;\; s.t.}$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Formulate the packing (dual) LP: Maximize profit

(Note: dynamic LP!)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do
   $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$
   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Primal and Dual

Online Access Control (3)

$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $\boxed{\begin{aligned} A_j \cdot Y_j &\leq C \\ D_j \cdot Y_j &\leq \mathbf{1} \\ Y_j &\geq \mathbf{0} \end{aligned}}$
(I)	(II)

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

s.t. constraints

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Online Access Control (3)

Primal and Dual

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

primal-dual framework

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Online Access Control (3)

Primal and Dual

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure) ← optimal embedding!
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do
   $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$
   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Online Access Control (3)

Primal and Dual

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$	$\max B_j^T \cdot Y_j \;\; s.t.$
$Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$	$A_j \cdot Y_j \leq C$
$X, Z_j \geq \mathbf{0}$	$D_j \cdot Y_j \leq \mathbf{1}$
	$Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Embedding cost vs profit?

Online Access Control (3)

Primal and Dual

$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$
(I)	(II)

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do
   $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$
   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

If cheap: accept and update primal variables (always feasible solution)

Primal and Dual

Online Access Control (3)

(I)	(II)
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Else reject

Primal and Dual

Online Access Control (3)

$$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$$
$$Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$$
$$X, Z_j \geq \mathbf{0}$$

(I)

$$\max B_j^T \cdot Y_j \;\; s.t.$$
$$A_j \cdot Y_j \leq C$$
$$D_j \cdot Y_j \leq \mathbf{1}$$
$$Y_j \geq \mathbf{0}$$

(II)

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do
   $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$
   (c) $z_j \leftarrow b_j - \gamma(j,\ell)$.
3. Else, (reject)
   (a) $z_j \leftarrow 0$.

Computationally hard!

Online Access Control (3)

Primal and Dual

$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$
(I)	(II)

Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

Does not know t'>t.
Competitive ratio:
r = Cost(ON)/Cost(OFF)

Algorithm

Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the jth round:

1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$ then, (accept)
   (a) $y_{j,\ell} \leftarrow 1.$
   (b) For each row e : If $A_{e,(j,\ell)} \neq 0$ do

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j,\ell).$
3. Else, (reject)
   (a) $z_j \leftarrow 0.$

Computationally hard!

Use your favorite approximation algorithm! If competitive ratio ρ and approximation r, overall competitive ratio ρ*r.

Reading / Literature Pointer

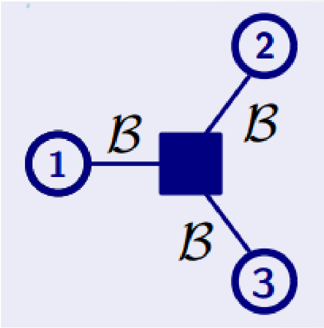
- Competitive and Deterministic Embeddings of Virtual Networks
  Guy Even, Moti Medina, Gregor Schaffrath, and Stefan Schmid.
  Journal Theoretical Computer Science (**TCS**), Elsevier, 2013.

- It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities
  Matthias Rost, Stefan Schmid, and Anja Feldmann.
  28th IEEE International Parallel and Distributed Processing Symposium (**IPDPS**), Phoenix, Arizona, USA, May 2014.

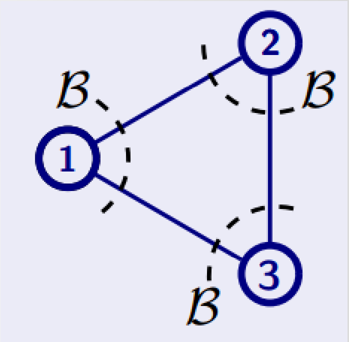
A Note on the Hose Model (1)

- Recall: Virtual Cluster Abstraction

- Two interpretations:
  - Logical switch at unique location
  - Logical switch can be distributed



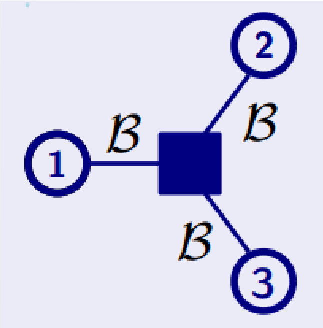
VS

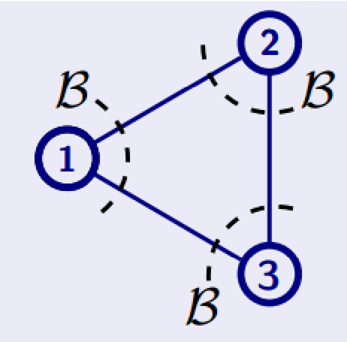


- If switch location unique
  - Polynomial-time algorithms: can try all locations…
  - … and then do our trick with the extra source.
  - What about Hose?

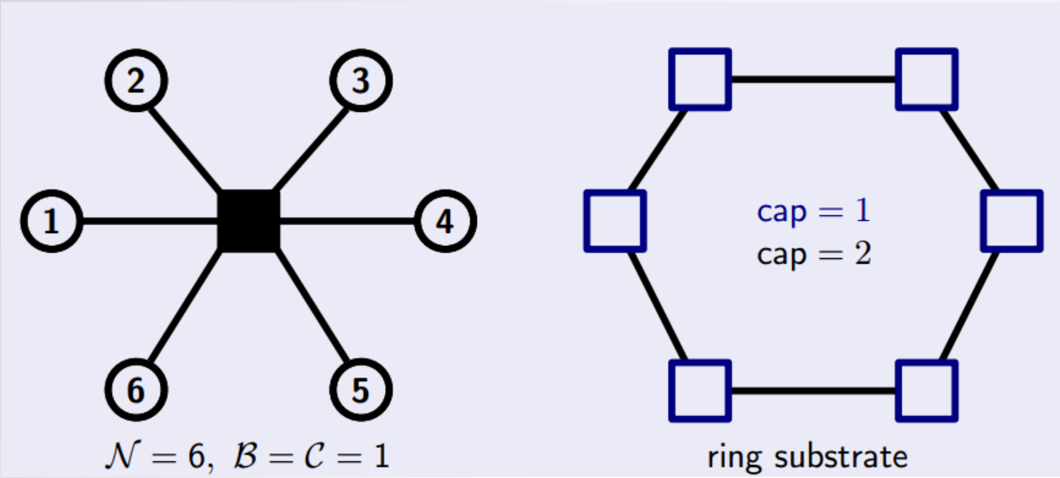
A Note on the Hose Model (2)

- Hose: More efficient?
- Deep classic result: The VPN Conjecture
  - In uncapacitated networks, hose embedding problems with symmetric bandwidth bounds and no restrictions on routing (SymG), can be reduced to hose problem instances in which routing paths must form a tree (known as the SymT model).
- Otherwise it can improve embedding footprint!
  - But is generally hard to compute



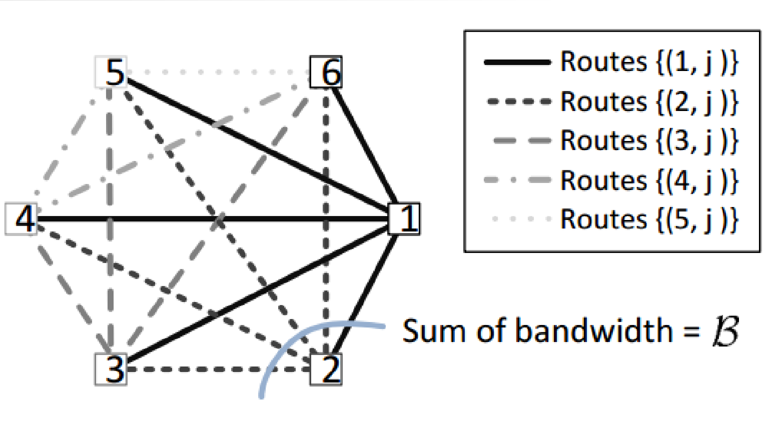


On the Benefit of Hose (1)



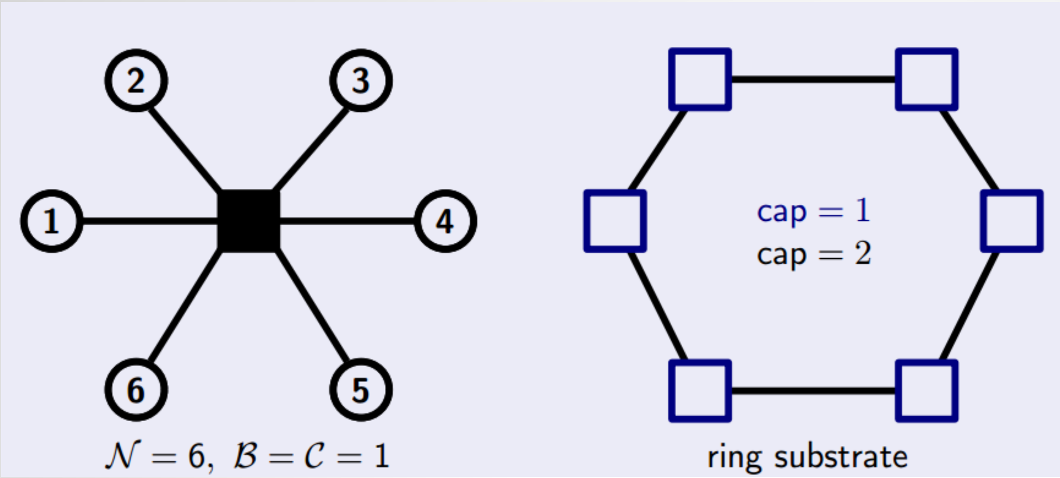
- VC: Compute and bandwidth one unit
- Substrate: compute one unit, links two units

- VC Request



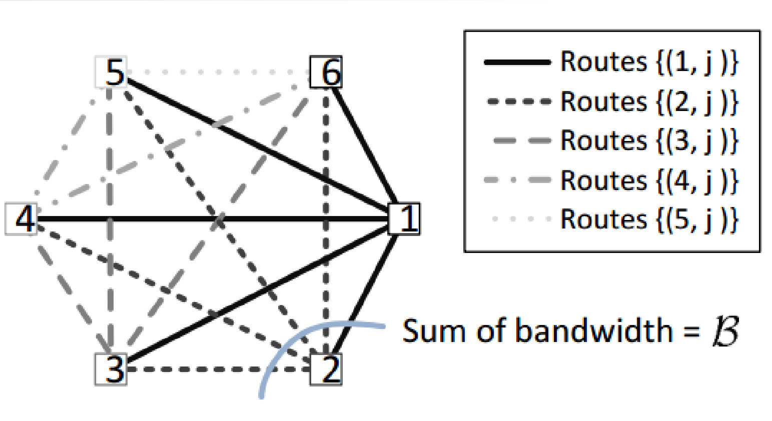
Thanks to Matthias Rost

On the Benefit of Hose (1)



- VC: Compute and bandwidth one unit
- Substrate: compute one unit, links two units

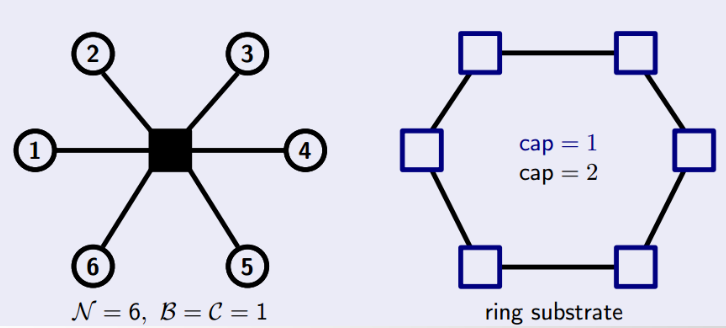
- VC Request



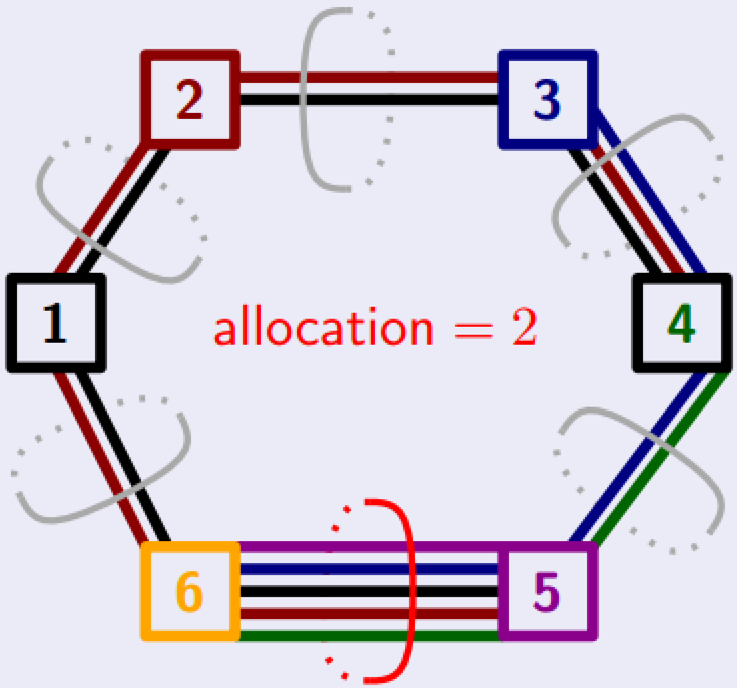
Impossible to map without splitting: need at least 5 independent paths to location where center is mapped!

Thanks to Matthias Rost

On the Benefit of Hose (2)



- In Hose model, it works!



Embedding on the same substrate

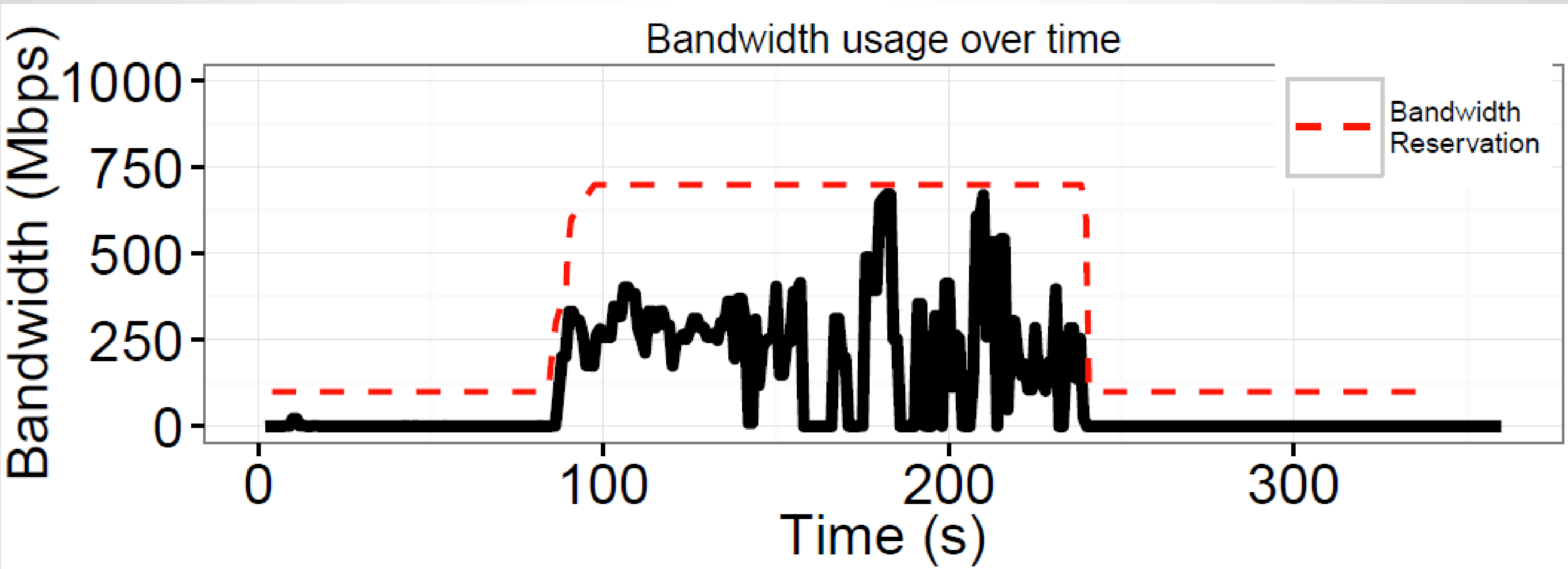
Why allocations of 2 are sufficient:

- Consider edge e between VMs 6 and 5.
- The edge is used by routes $R(e) = \{(1,5),(2,5),(3,6),(4,6),(5,6)\}$.
- Any valid traffic matrix M will respect:
  - $M_{1,5} + M_{2,5} \leq 1$
  - $M_{3,6} + M_{4,6} + M_{5,6} \leq 1$
- Hence $\sum_{(i,j)\in R(e)} M_{i,j} \leq 2$ holds.

Thanks to Matthias Rost

The need for adjustments

Constant reservations would be wasteful:



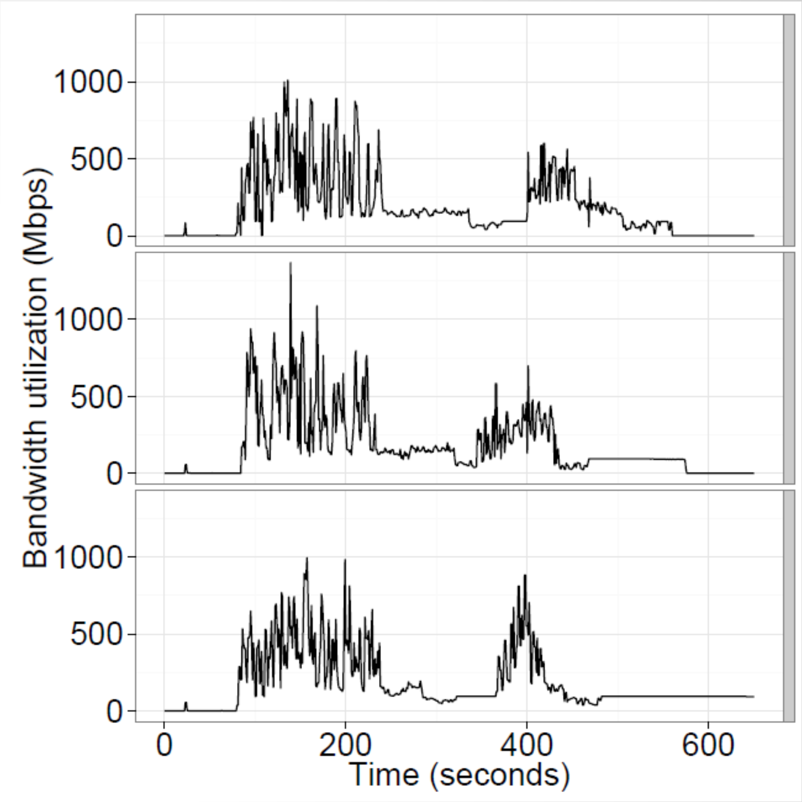
Bandwidth utilization of a TeraSort job over time.

In red: good bandwidth reservation.

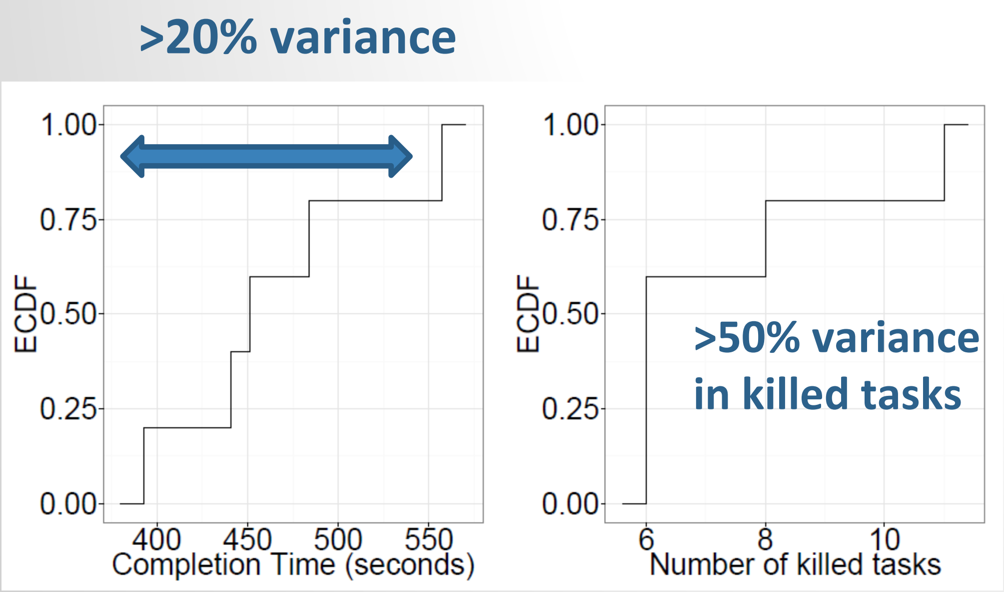
(Tasks inform Hadoop controller prior to shuffle phase; reservation with Linux *tc*.)

The need for online adjustments

- ***Temporal*** resource patterns are hard to predict
- Resource allocations must be changed ***online***



Bandwidth utilization of 3 different runs of the same **TeraSort workload** (**without interference**)

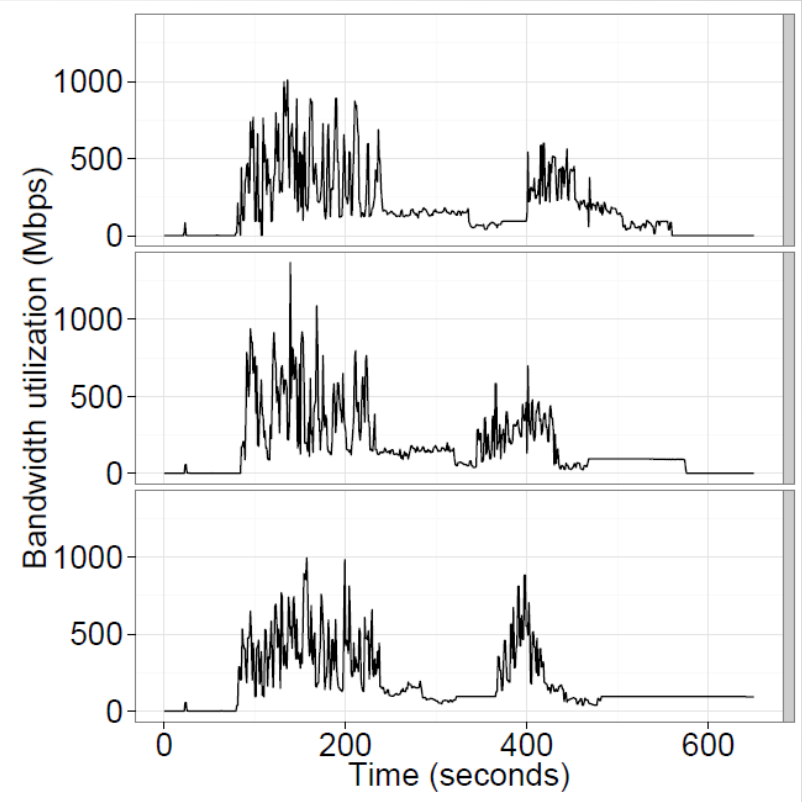


Completion times of jobs in the presence of **speculative execution** (*left*) and the number of speculated tasks (*right*)

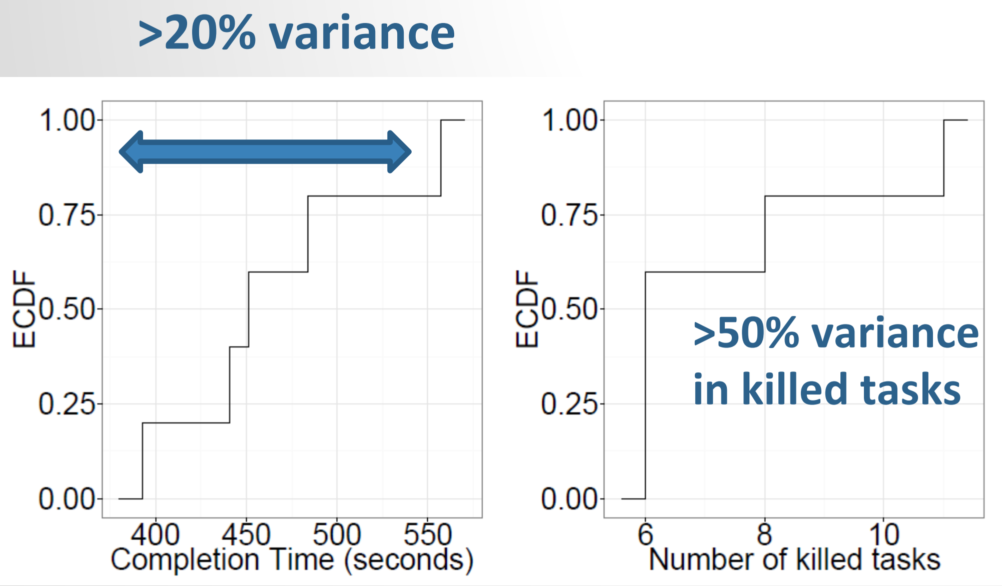
The need for online adjustments

- ❑ ***Temporal*** resource patterns are hard to predict

- ❑ Resource allocations must be changed ***online***



Bandwidth utilization of 3 different runs of the same **TeraSort workload** (**without interference**)

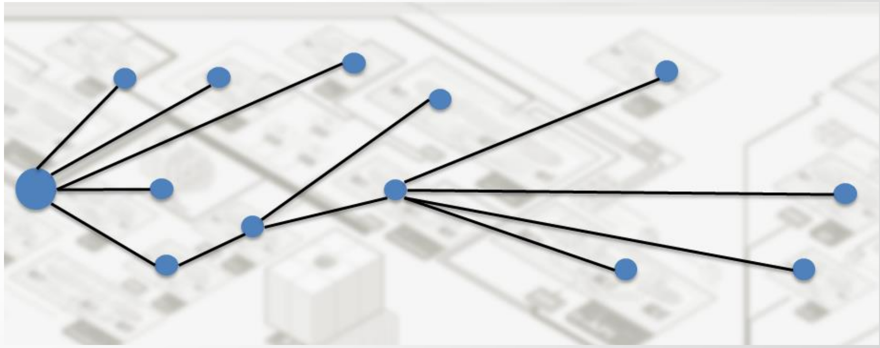


Completion times of jobs in the presence of **speculative execution** (*left*) and the number of speculated tasks (*right*)

Latency-Critical Applications

- Another critical requirement besides bandwidth, especially in cloud data stores is ***latency***
  - Today's interactive **web** applications require **fluid** response time
  - Degraded user experience directly impacts **revenue**

- **Tail** matters...
  - Web applications = multi-tier, **large** distributed systems
  - 1 request involves **10(0)s** data accesses / servers!
  - A **single late** read may delay entire request



How to cut tail latency?

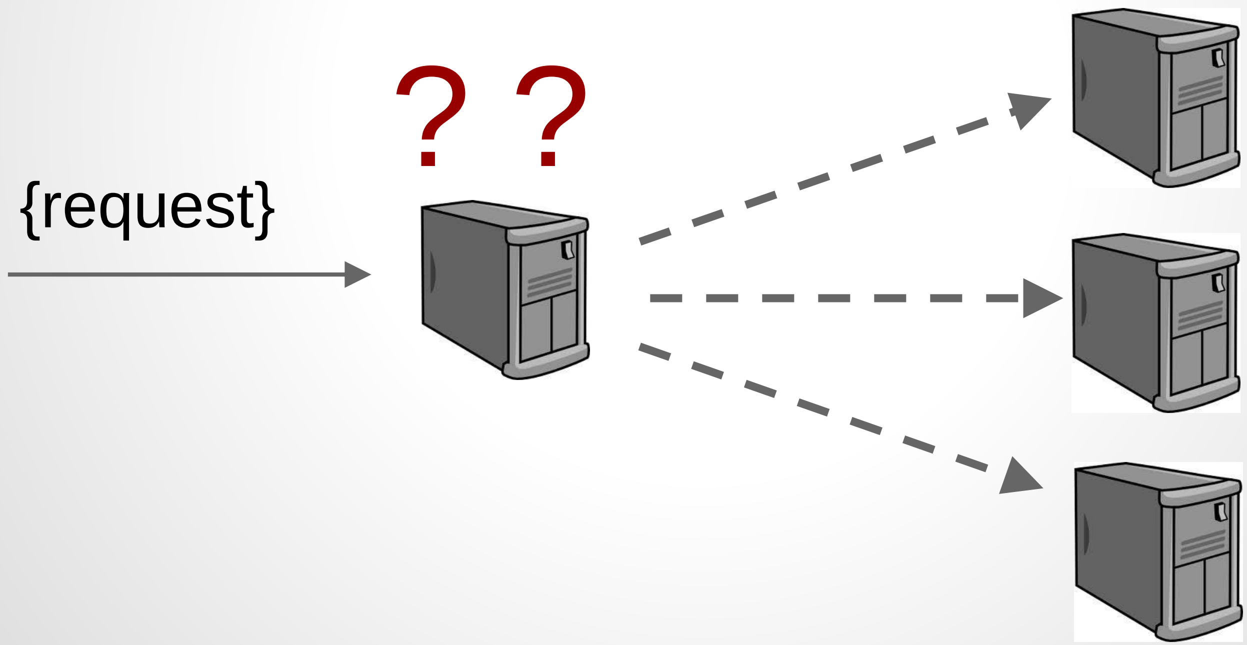
- How to guarantee low tail in shared cloud? A non-trivial challenge even in **well-provisioned** systems
  - **Skews** in demand, time-varying service times, stragglers, ...
  - No time to make make **rigorous optimizations or reservations**

- Idea: Exploit **replica selection**!
  - Many distributed DBs resp. **key-value stores** have redundancy
  - **Opportunity** often overlooked so far

- Our focus: **Cassandra** (1-hop DHT, server = client)
  - Powers, e.g., Ebay, Netflix, Spotify
  - More sophisticated than MongoDB or Riak

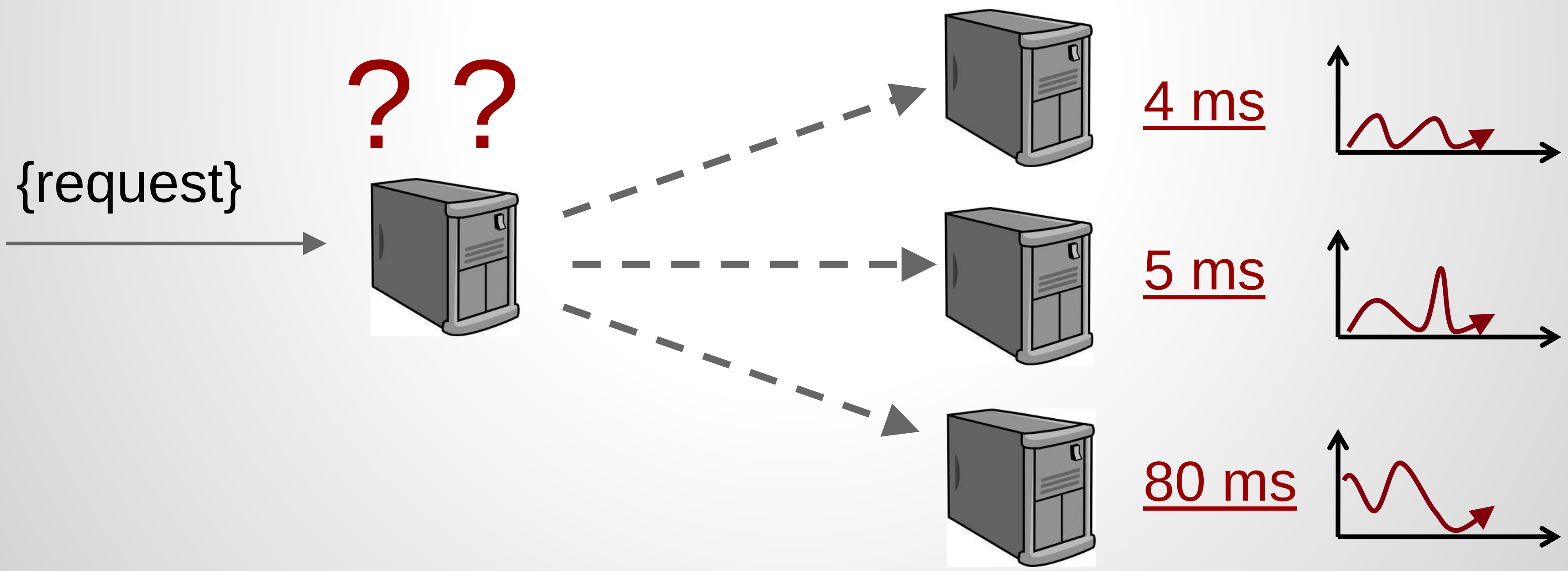
Exploit Replica Selection

- Great idea! But how? Just go for «the best»?



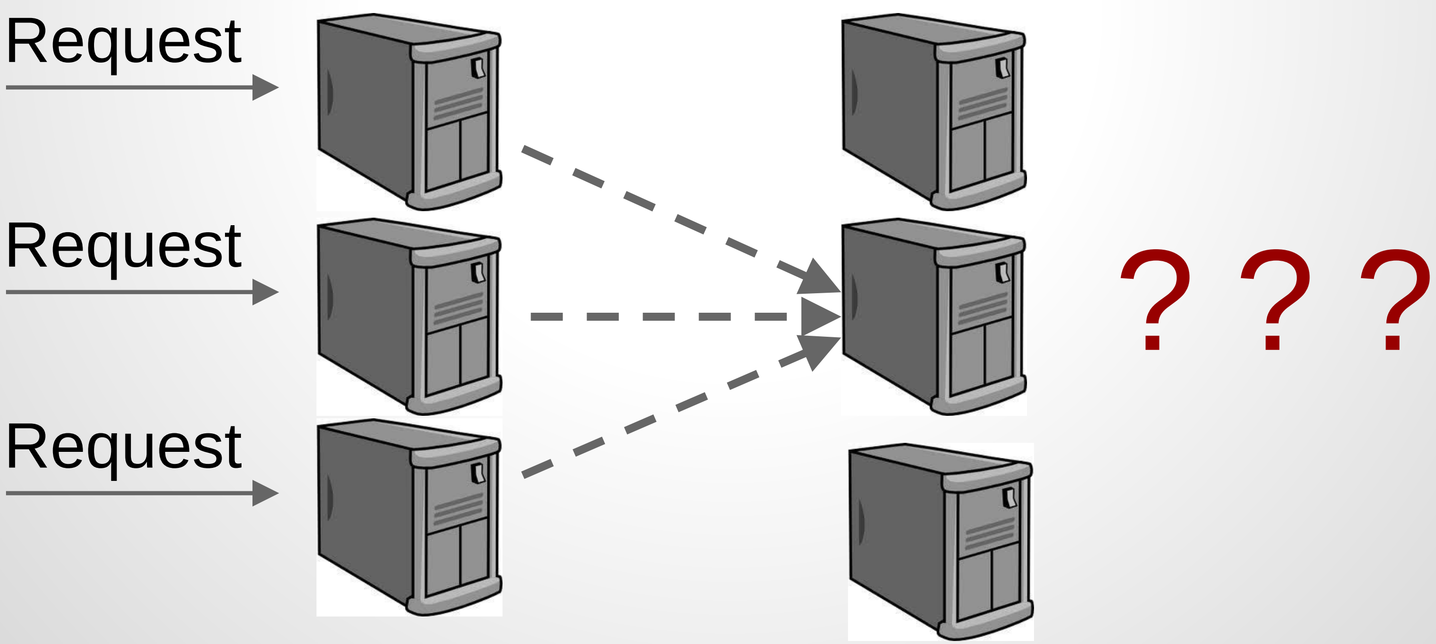
Careful: «The best» can change

- Not so simple!
  - Need to deal with **heterogenous** and **time-varying** service times
  - Background garbage collection, log compaction, TCP, deamons



Careful: Herd Behavior

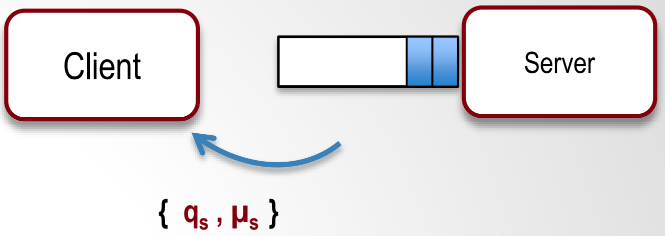
- Potentially high **fan-in** and **herd behavior**!
- Observed in Cassandra Dynamic Snitching (DS)
  - Coarse **time intervals** and **I/O gossiping**
  - **Synchronization** and stale information



A coordination / control theory problem!

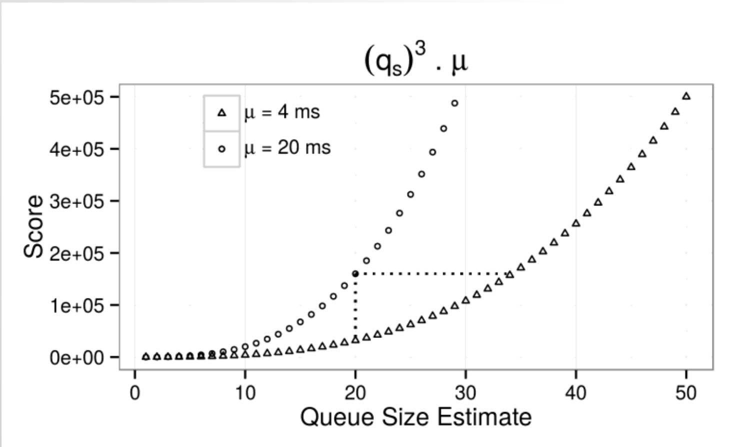
C3 in a Nutshell

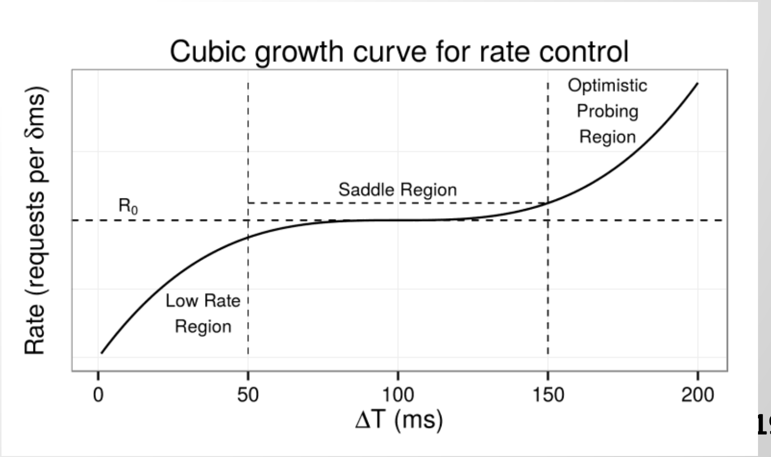
- 4 Principles:
  - Stay informed: **piggy-back** queue state and service times
  - Stay reactive and don't commit: use **backpressure queue**
  - Leverage heterogeity: **compensate** for service times
  - Avoid redundancy



- Mechanism 2: rate control
  - Goal: match service rate and keep pipeline full
  - Cubic, with saddle region

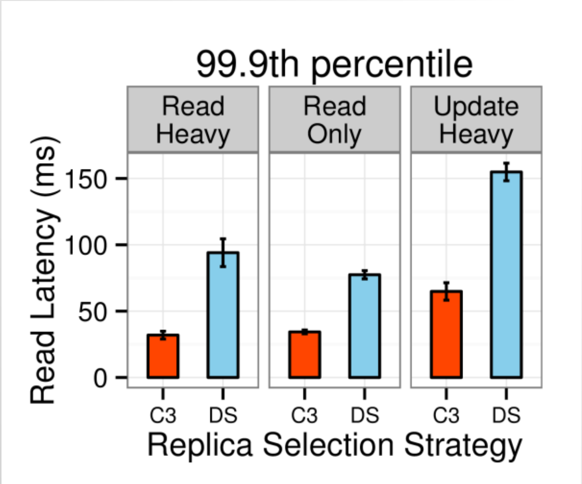
- Mechanism 1: replica ranking
  - Penalize larger queues



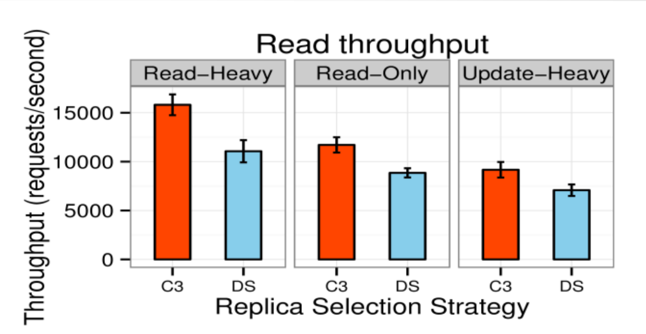


Performance Evaluation

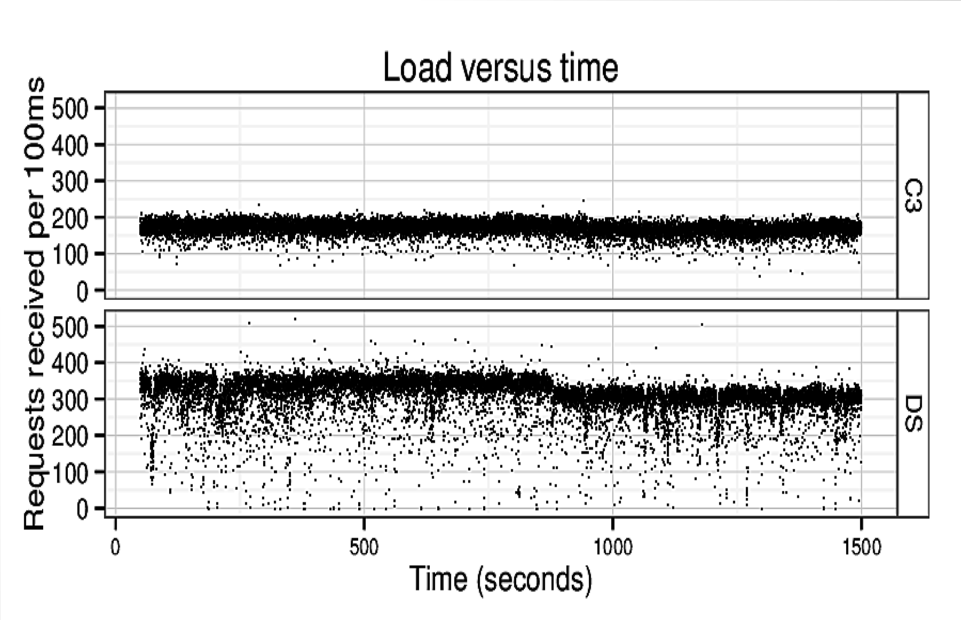
- Methodology:
  - **Amazon EC2**
    - disk vs SSD
  - BigFoot testbed
  - Simulations
- Lower tail latency
  - 2-3x for 99.9%



- Higher read throughput...



- ... and lower load (and variance)!



Reading / Literature Pointer

- C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection
  Lalith Suresh, Marco Canini, Stefan Schmid, and Anja Feldmann.
  12th USENIX Symposium on Networked Systems Design and Implementation (**NSDI**), Oakland, California, USA, May 2015.

Conclusion

- Virtualization opens many flexibilities for resource allocation
- Underlying computational problems often hard, but not always!
- Online Primal-Dual Framework can give guarnatees over time
- What are the threats of offering such new services?
- How to deal with more general specifications

Thank you!

Some Own Related Work

Beyond the Stars: Revisiting Virtual Cluster Embeddings
Matthias Rost, Carlo Fuerst, and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), July 2015.

Online Admission Control and Embedding of Service Chains
Tamás Lukovszki and Stefan Schmid.
22nd International Colloquium on Structural Information and Communication Complexity (**SIROCCO**), Montserrat, Spain, July 2015.

How Hard Can It Be? Understanding the Complexity of Replica Aware Virtual Cluster Embeddings
Carlo Fuerst, Maciek Pacut, Paolo Costa, and Stefan Schmid.
23rd IEEE International Conference on Network Protocols (**ICNP**), San Francisco, California, USA, November 2015.

Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks
Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, and Anja Feldmann.
USENIX Annual Technical Conference (**ATC**), Philadelphia, Pennsylvania, USA, June 2014.

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.

Tutorial

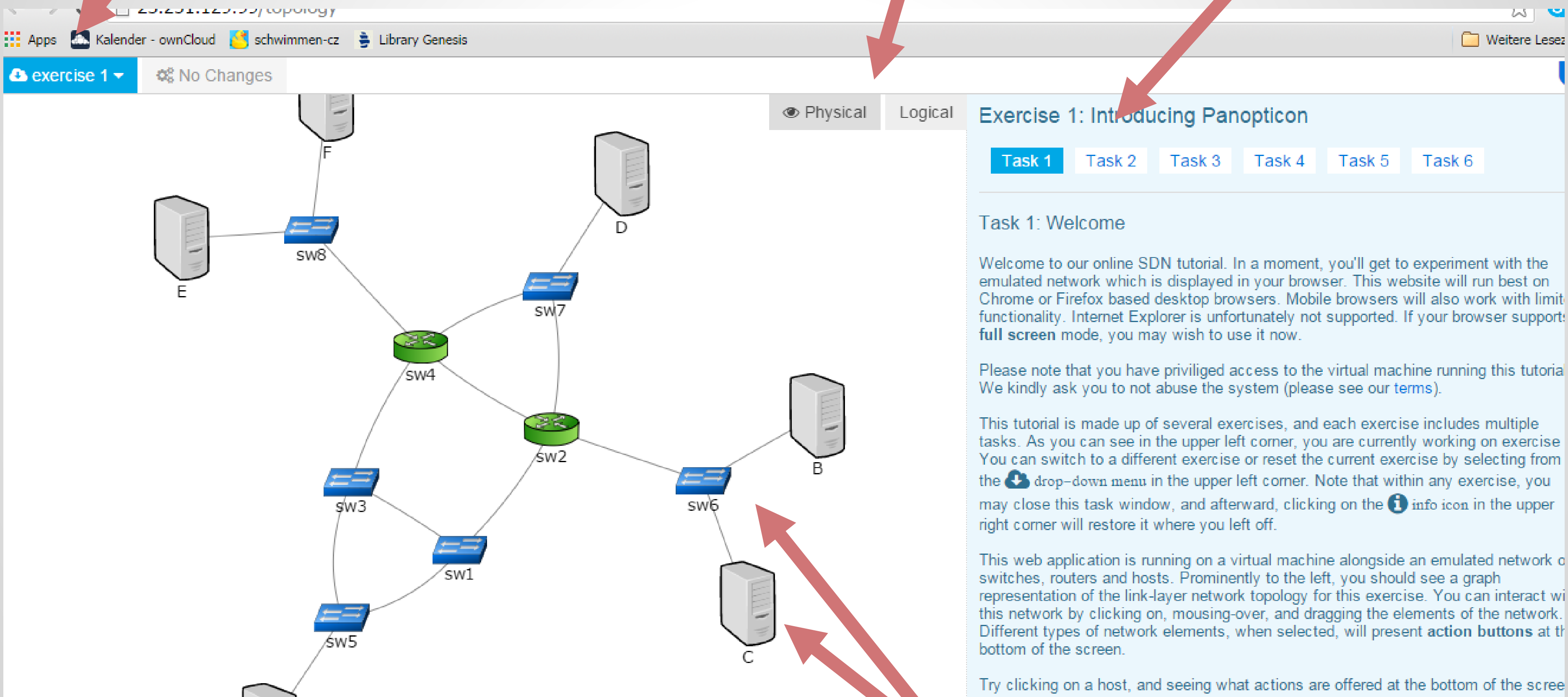
- Understanding hybrid SDN deployments
- Tutorial based on USENIX ATC 2014 paper on Panopticon

Overview: workshop@stacktile.io

Exercise

Physical / Logical View

Task



Shell

The 3 Hands-On Exercises

- Exercise 1:
  - Learning about waypoint enforcement and isolation concepts
  - Setting access control policies

- Exercise 2:
  - Dealing with layer-3 devices (routers, WPE across subnet boundaries)
  - Supporting migration (IP subnet mobility)

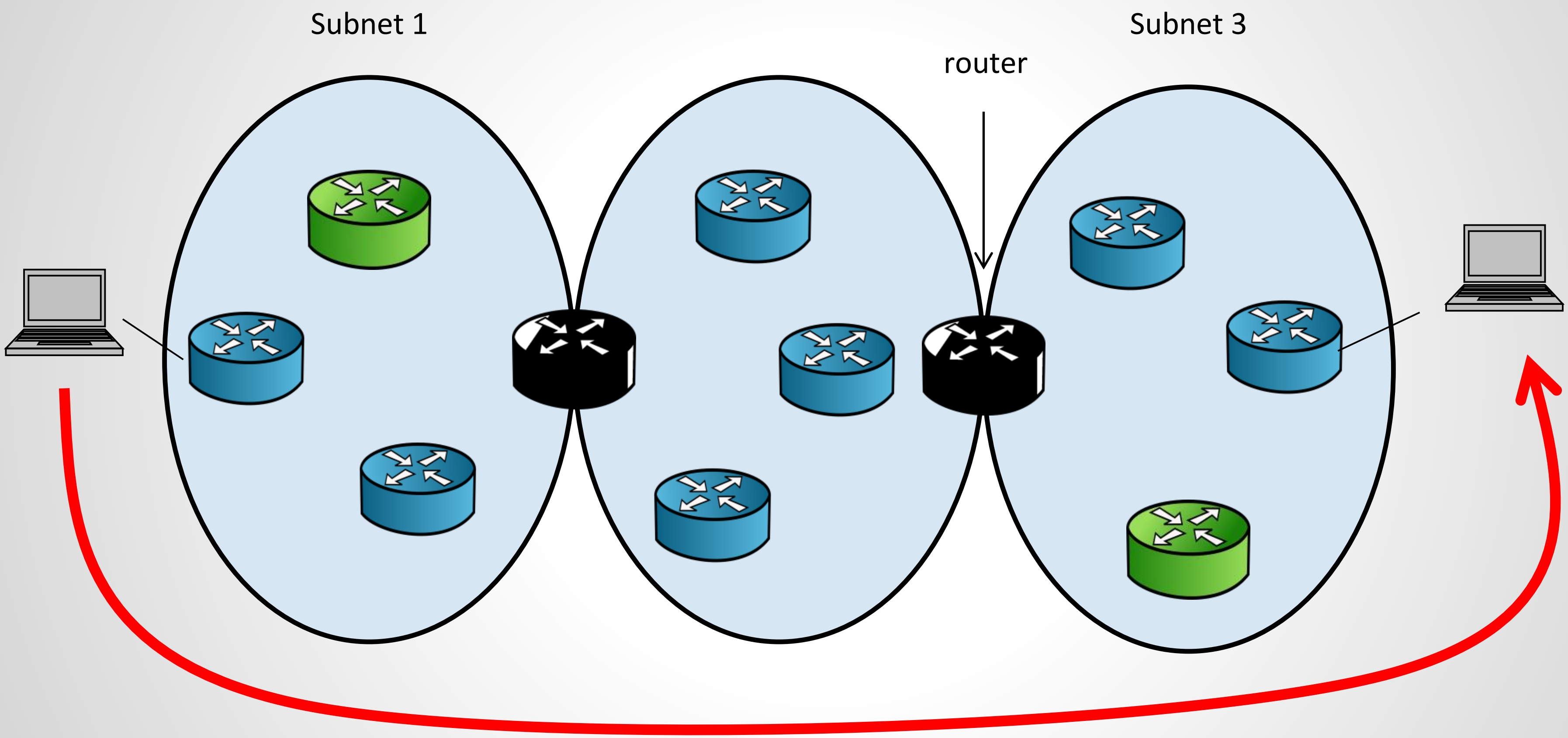
- Exercise 3:
  - Dealing with layer-3 devices (routers)
  - Supporting migration

Exercise 1: Discussion

- All hosts in same IP subnet
- But different VLANs
- Hosts connected to all reachable SDN switches via SDNc port / VLAN
- One SDN switch on border is designated one
- Via Waypoint Enforcement: Can perform access control!

Exercise 2: Discussion (1)

- Mobility between subnets with an SDN router



Today: complex MPLS tunnels.
Alternative with SDN: logically centralized.

Exercise 2: Discussion (1)

- Waypoint Enforcement even if SDNc port communicates with non-SDNc port
- IP encapsulation such as GRE can be used by SDN switches to tunnel traffic across IP subnet boundaries
- Can also enforce simple policies

Exercise 3: Discussion (1)

- Panopticon can be used to enforce middlebox traversal in hybrid SDN deployments
- Two virtualized linux iptables-based firewalls
- Panopticon ensures the properties of guaranteed waypoint enforcement across subnet boundaries in the presence of middleboxes in a simple IP routed network topology

Exercise 3: Discussion (2)