Performance and Security Isolation in Softwarized Networks: Advances and Challenges

Stefan Schmid

Faculty of Computer Science
University of Vienna

3rd Workshop on Advances in Slicing for Softwarized Infrastructures (S4SI 2020)

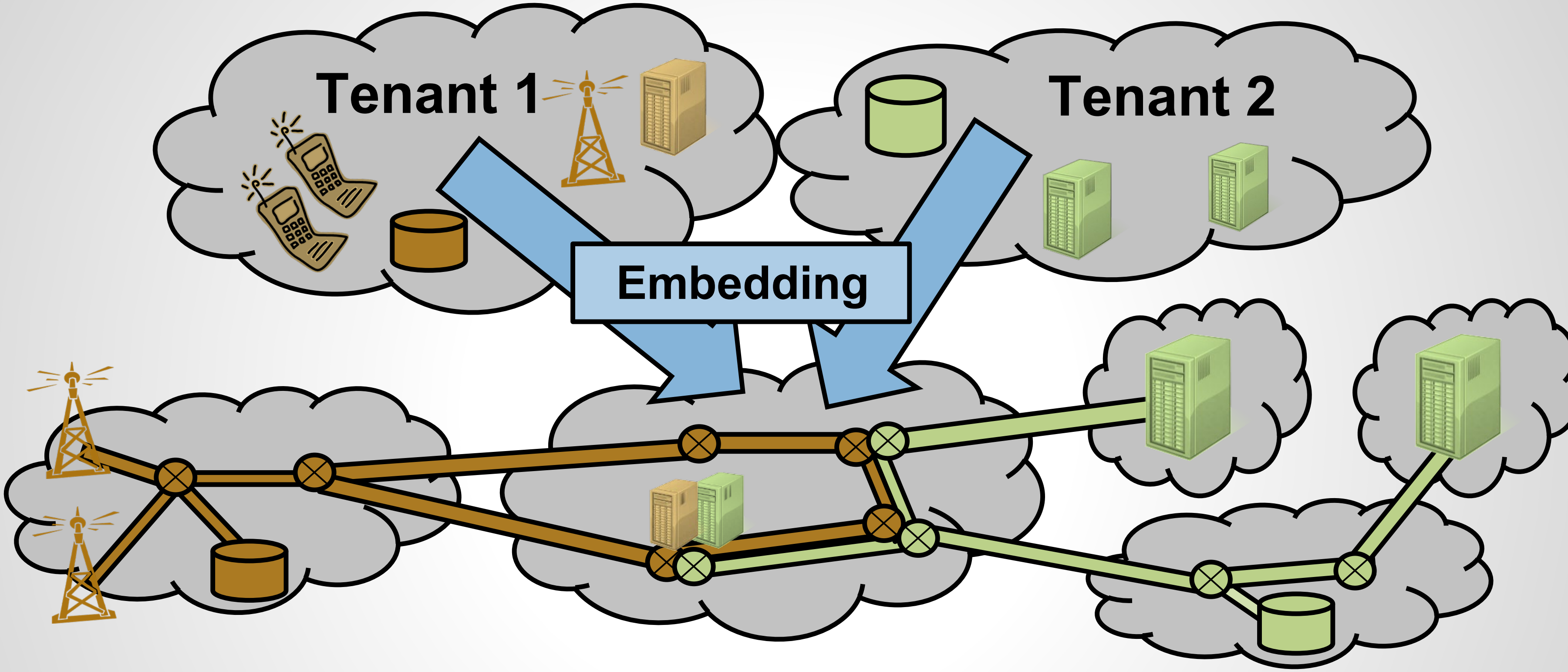
Softwarized Networks:
It's a great time to be a networking researcher!

Rhone and Arve Rivers, Switzerland

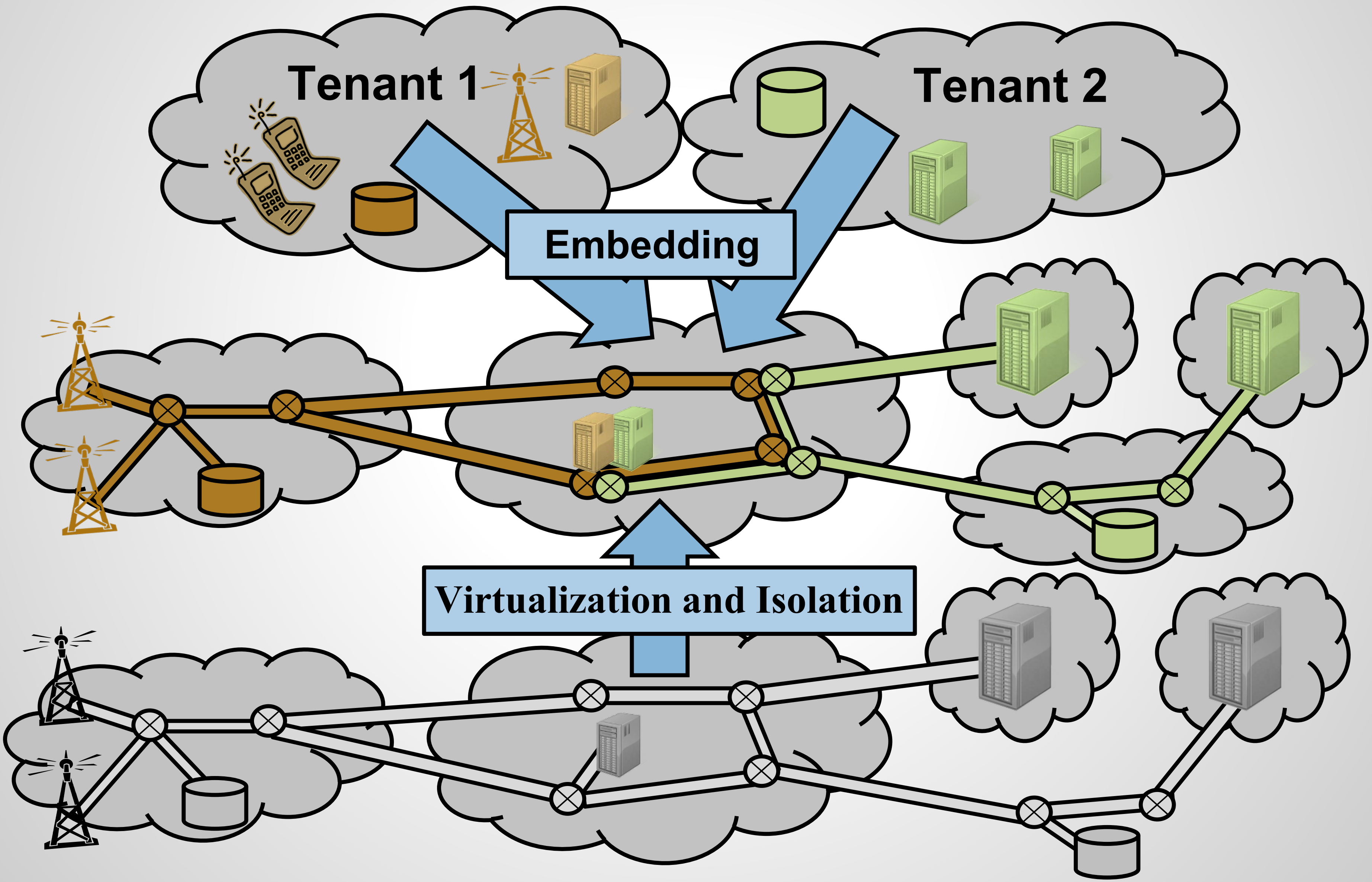
Credits: George Varghese.

The (At Least) 3 Dimensions of Network Flexibility

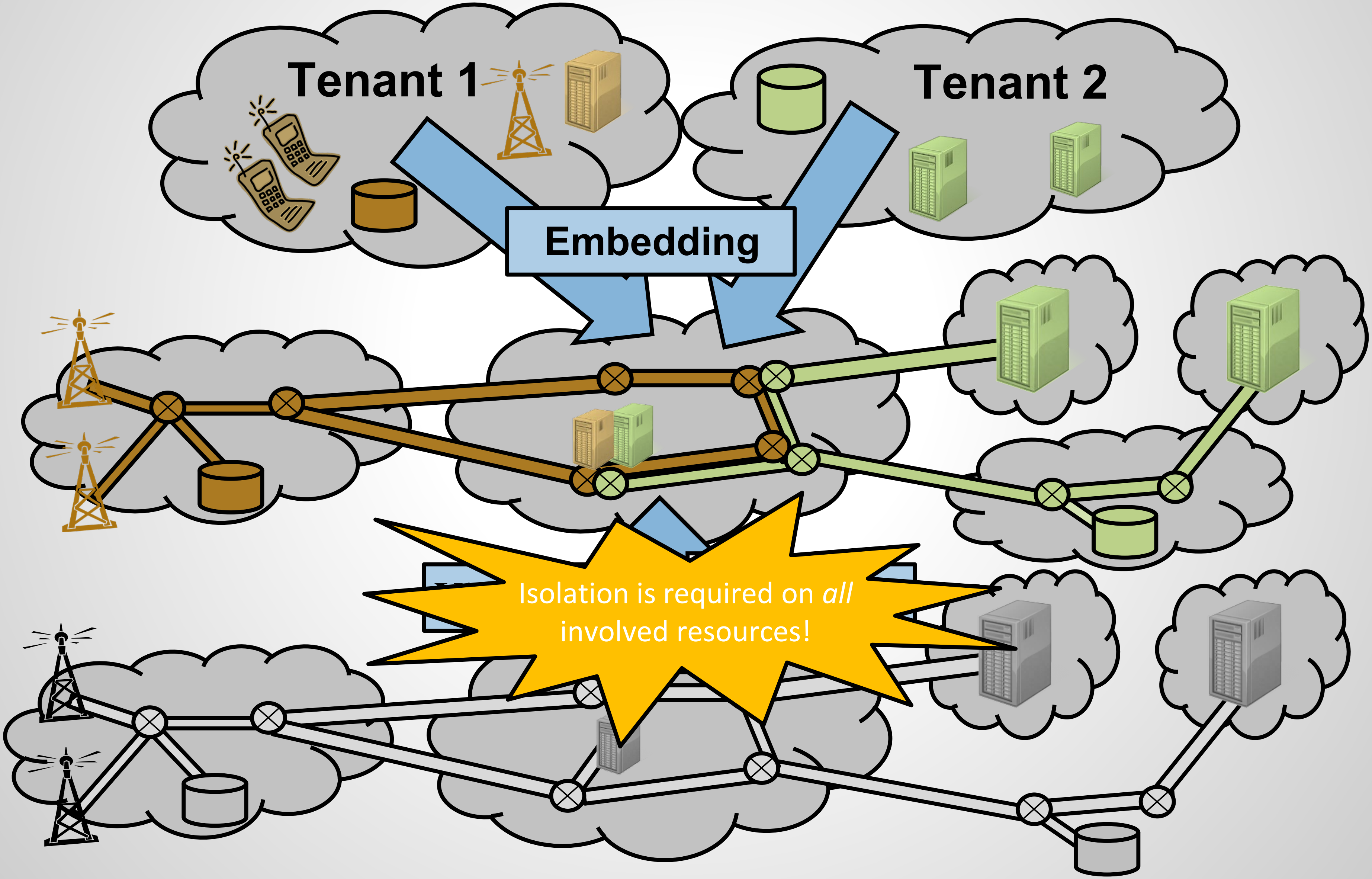
Opportunity: Improved Sharing of Physical Network Infrastructure



Challenges: Isolation (and Embedding)

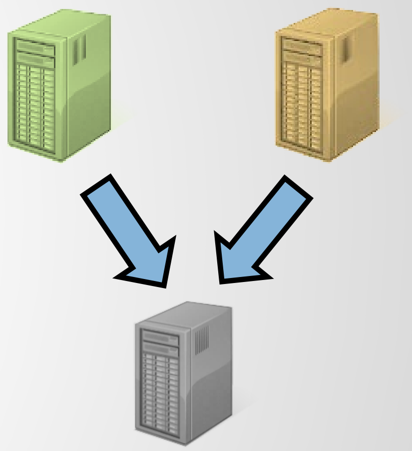


Challenges: Isolation (and Embedding)



Two Flavors of Isolation

- Logical isolation
  - E.g., prevent from communication, no need to coordiante name/address space, etc.
  - Relevant for security
- Performance isolation
  - E.g., prevent resource interference, ensure SLAs, make it appear like a dedicated infrastructure
  - Relevant for quality-of-service



We'll consider both in this talk!

Invitation: A Roadtrip Through The Opportunities and Challenges of Network Isolation

- ❑ Opportunities
  - ❑ Algorithmic opportunities
  - ❑ Technological opportunities
- ❑ Challenges
  - ❑ Modelling challenges
  - ❑ Security challenges
- ❑ A perspective how AI can improve slicing efficiency and security

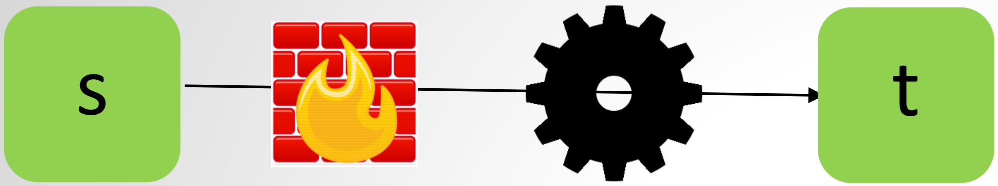
Road map 1927: Arizona and New Mexico

Invitation: A Roadtrip Through The Opportunities and Challenges of Network Isolation

- ❑ Opportunities
  - ❑ **Algorithmic opportunities**
  - ❑ Technological opportunities
- ❑ Challenges
  - ❑ Modelling challenges
  - ❑ Security challenges
- ❑ A perspective how AI can improve slicing efficiency and security

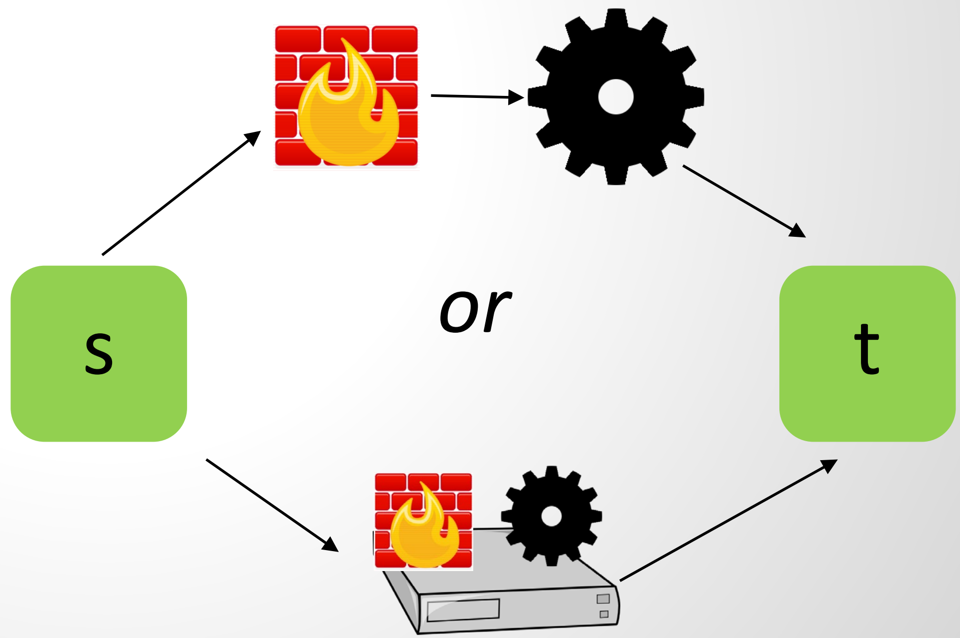
Road map 1927: Arizona and New Mexico

Opportunity: Define and Flexibly Allocate Complex Services

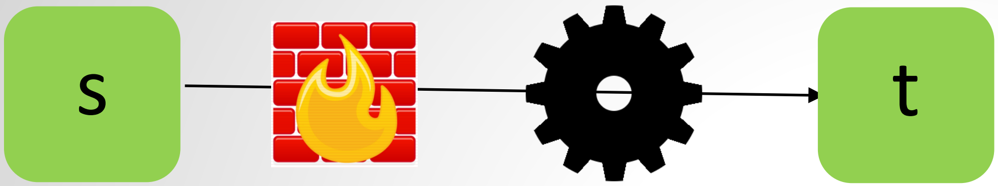


Steer traffic through network functions to compose complex service chains

More complex requests: allowing for alternatives and different decompositions

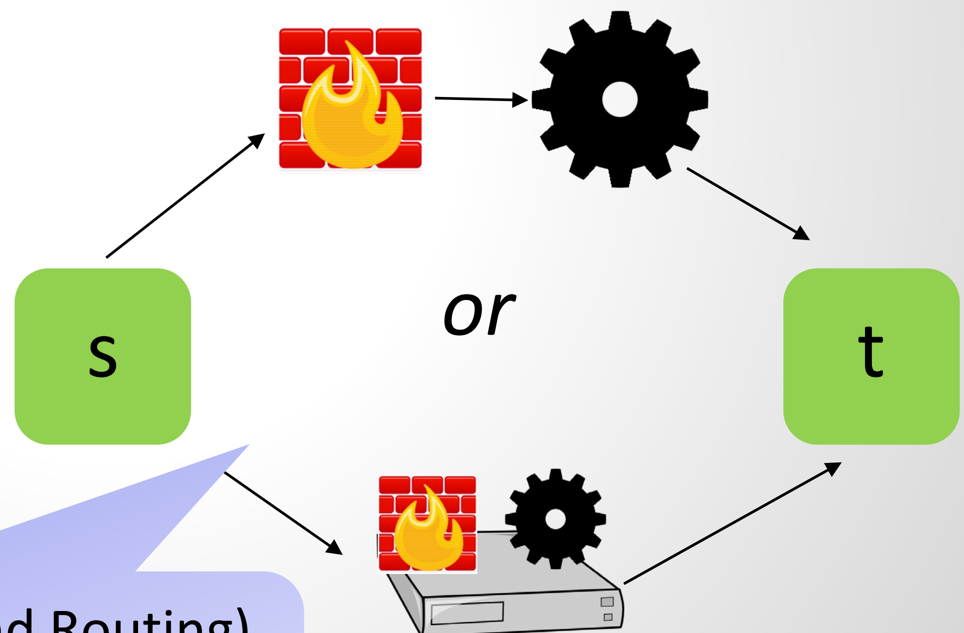


Opportunity: Define and Flexibly Allocate Complex Services



Steer traffic through network functions to compose complex service chains

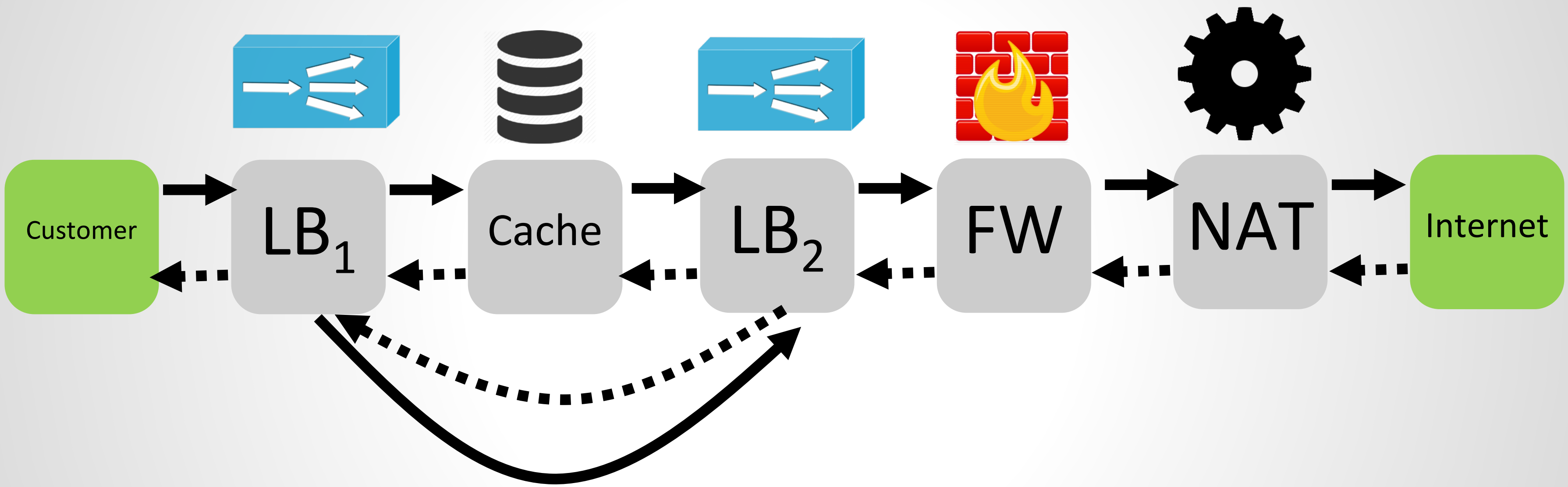
More complex requests: allowing for alternatives and different decompositions



Known as PR (Processing and Routing) Graph: allows to model different choices and implementations!

More Complex Service Chains

IETF Draft: Service chain for mobile operators

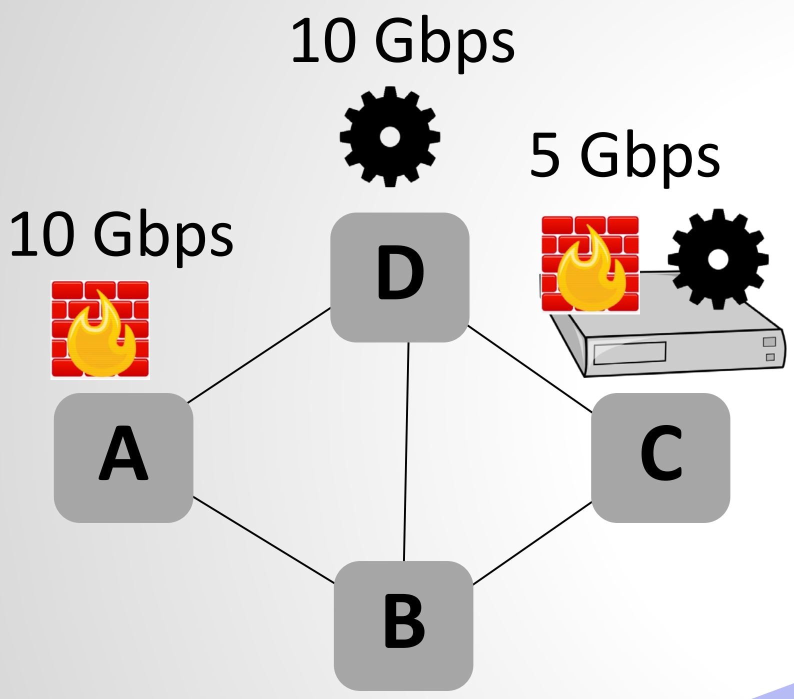


- ❑ Load-balancers are used to route (parts of) the traffic through cache

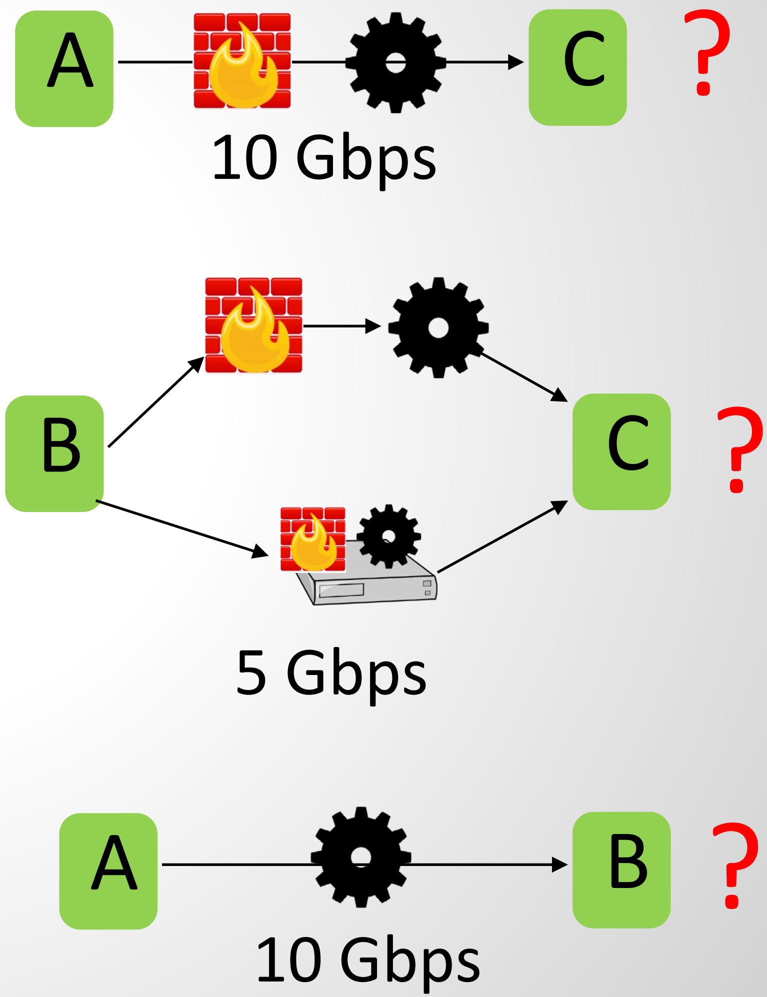
Source: https://tools.ietf.org/html/draft-ietf-sfc-use-case-mobility-06

Algorithmic Challenges: Admission Control and Embedding

Substrate:



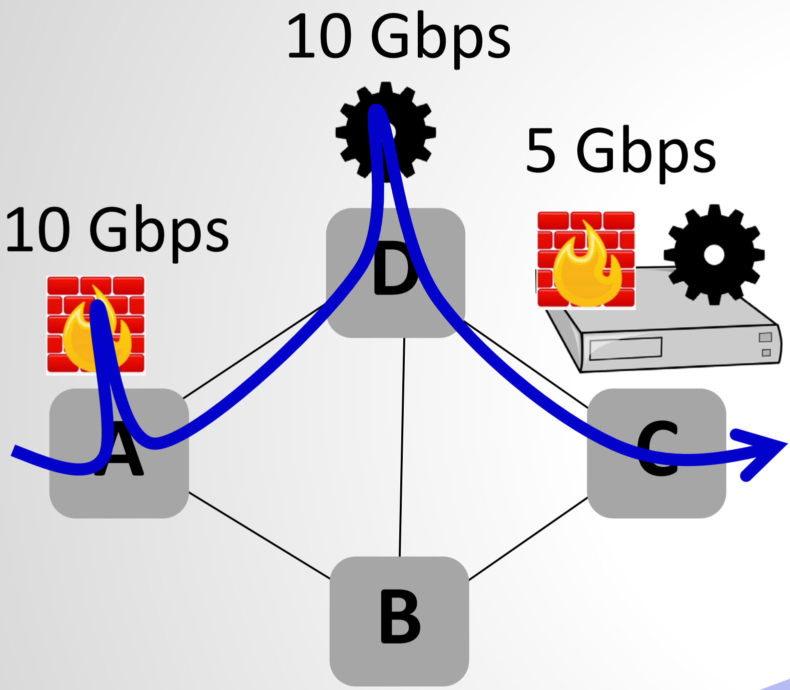
Requests:



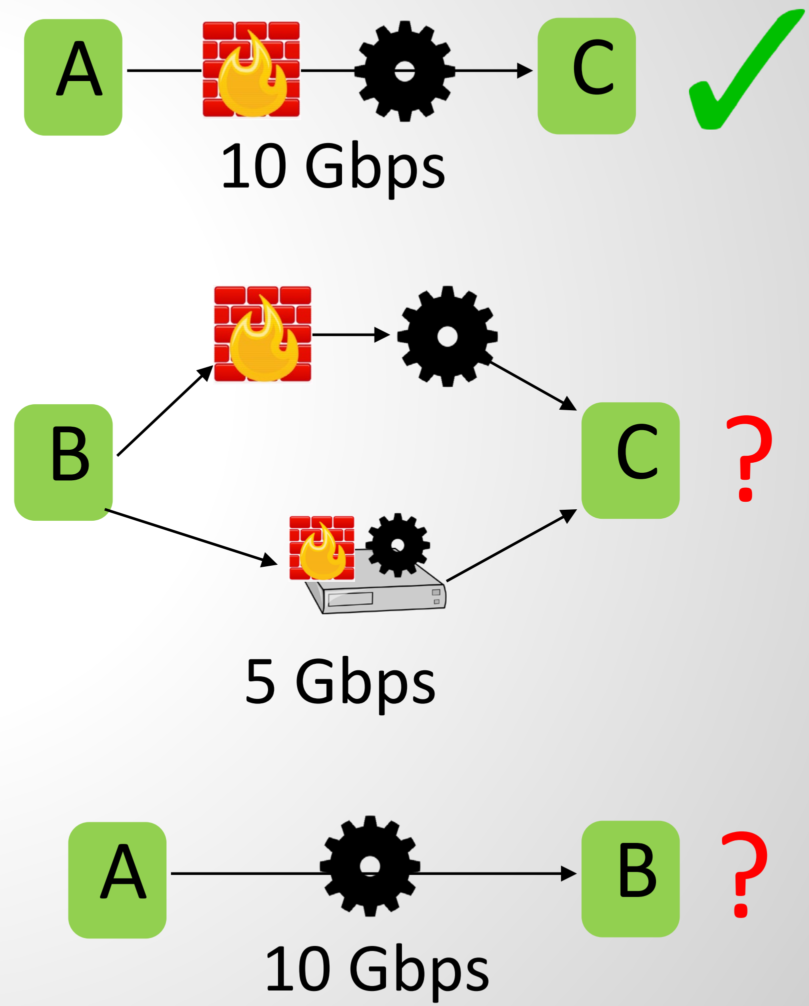
Which ones can be admitted and embedded?

Algorithmic Challenges: Admission Control and Embedding

Substrate:



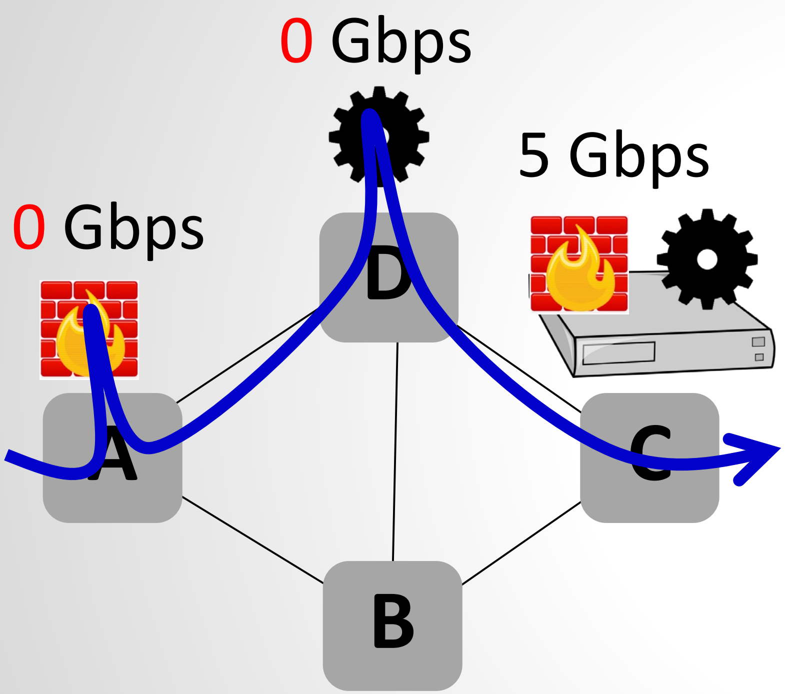
Requests:



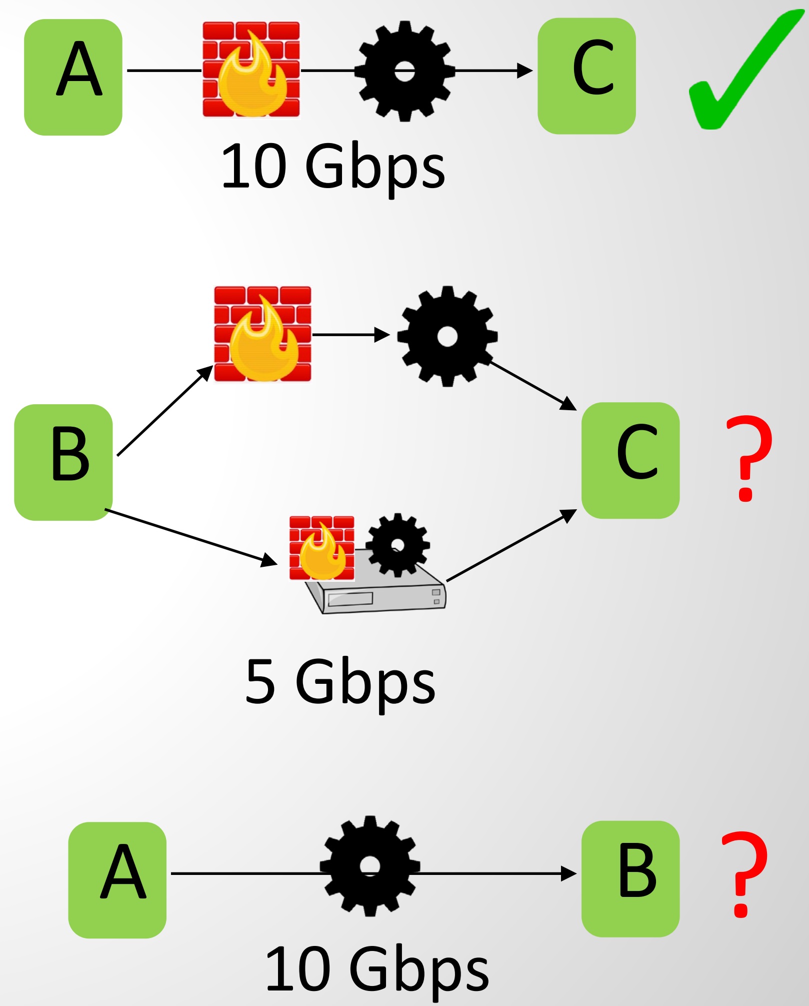
Which ones can be admitted and embedded?

Algorithmic Challenges:
Admission Control and Embedding

Substrate:



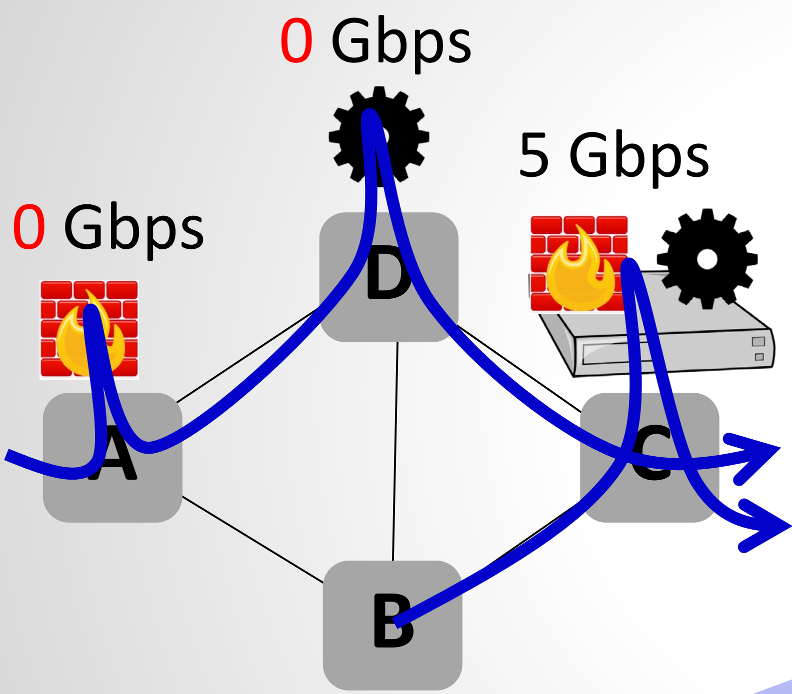
Requests:



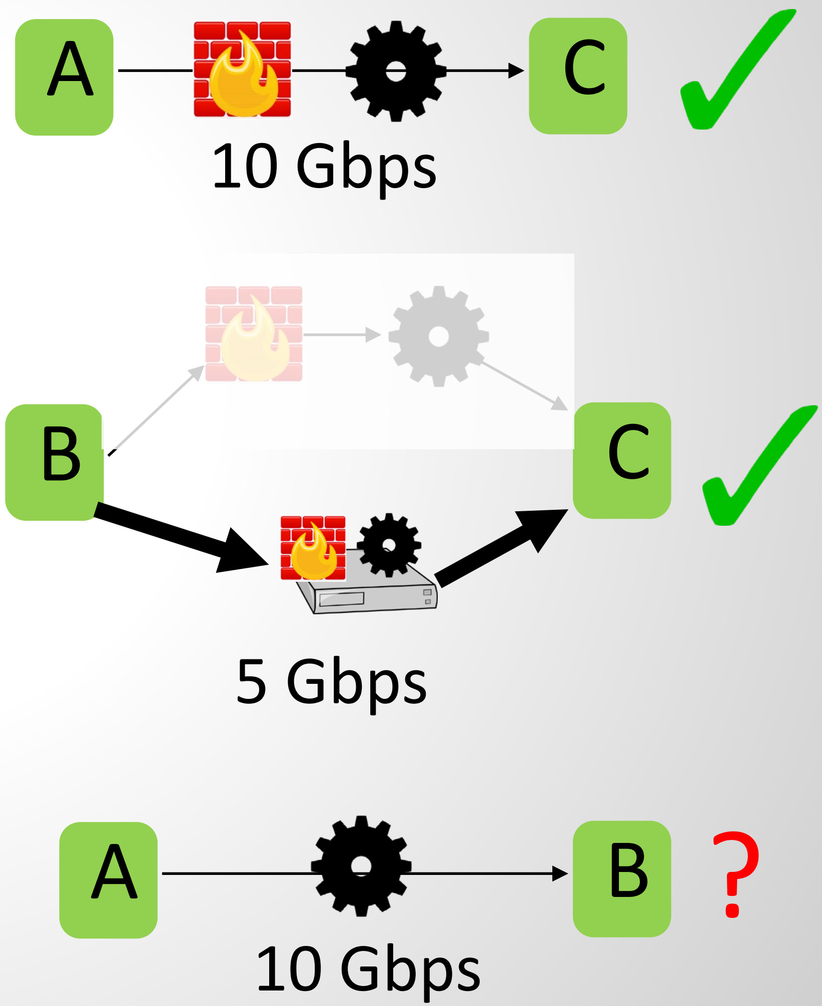
Deduct resources for performance isolation!

Algorithmic Challenges: Admission Control and Embedding

Substrate:



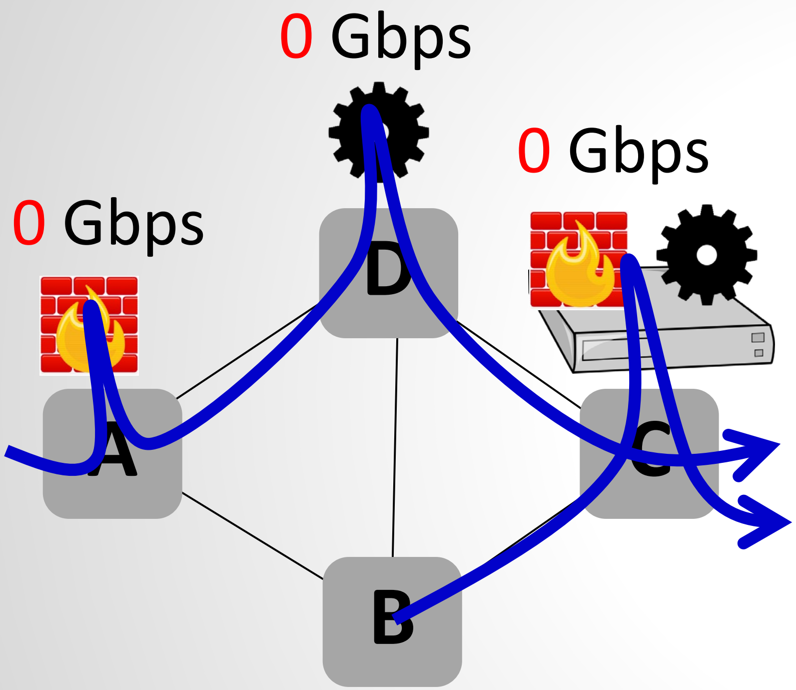
Requests:



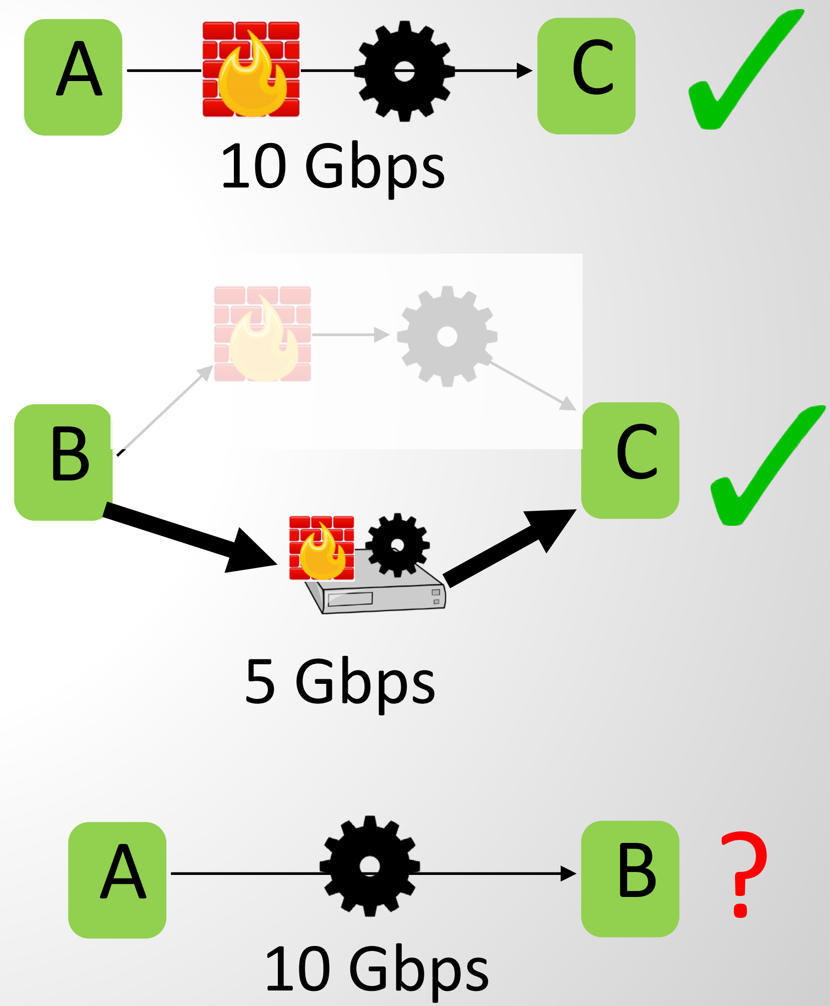
Which ones can be admitted and embedded?

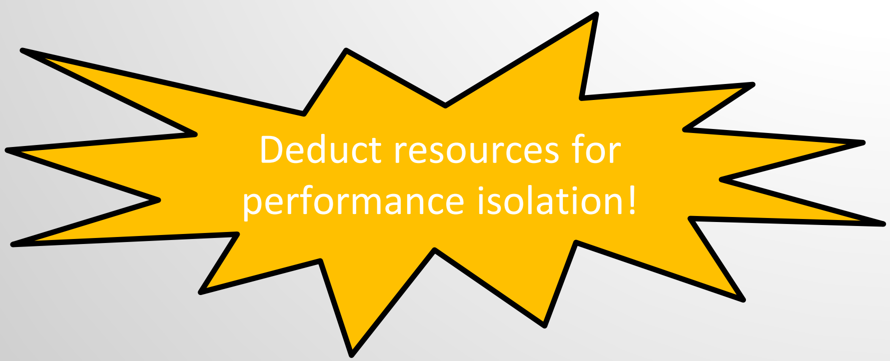
Algorithmic Challenges: Admission Control and Embedding

Substrate:



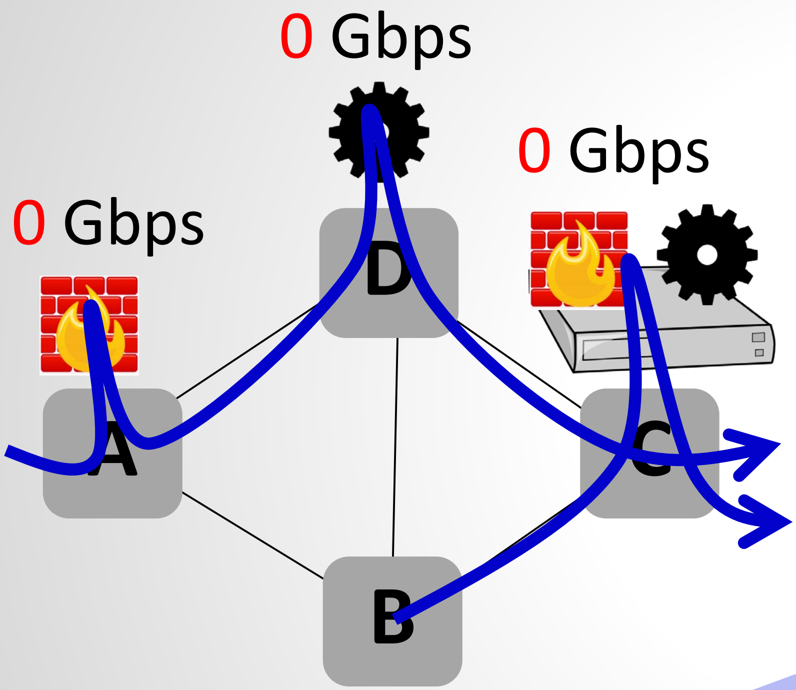
Requests:



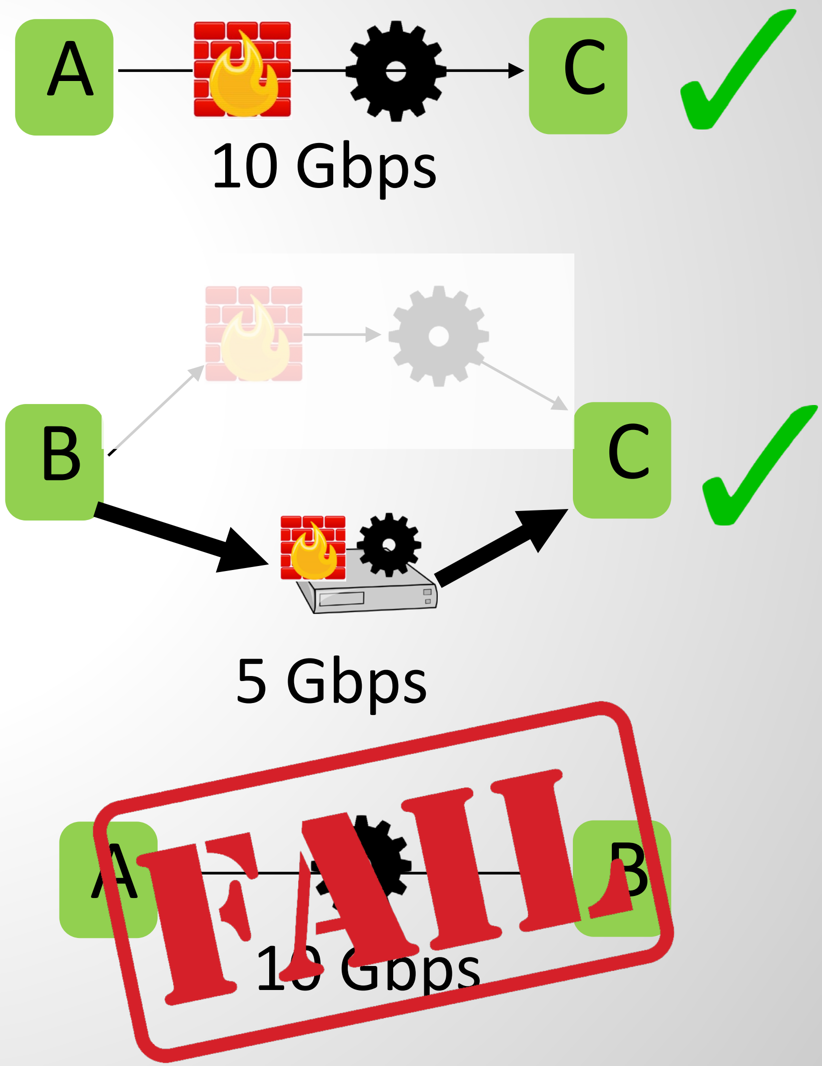


Algorithmic Challenges: Admission Control and Embedding

Substrate:



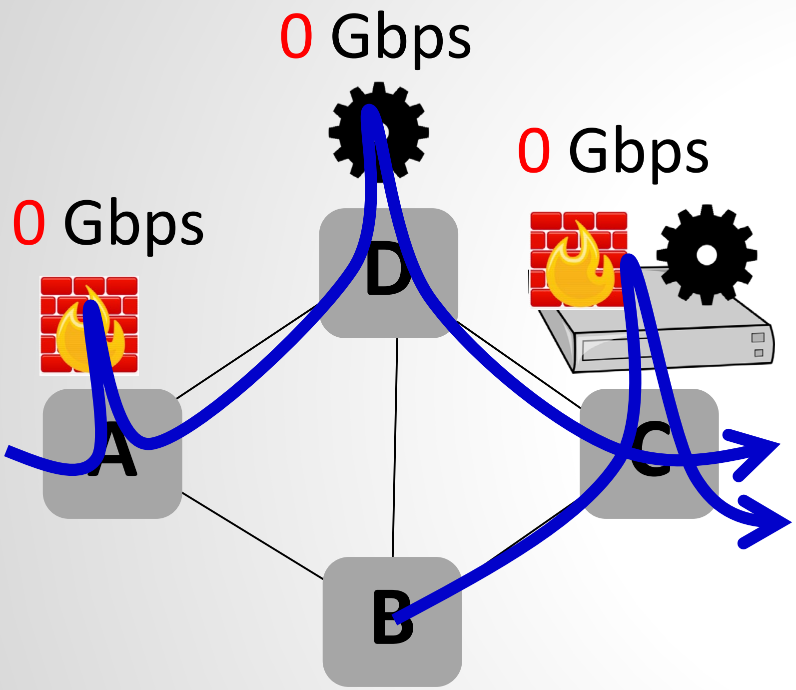
Requests:



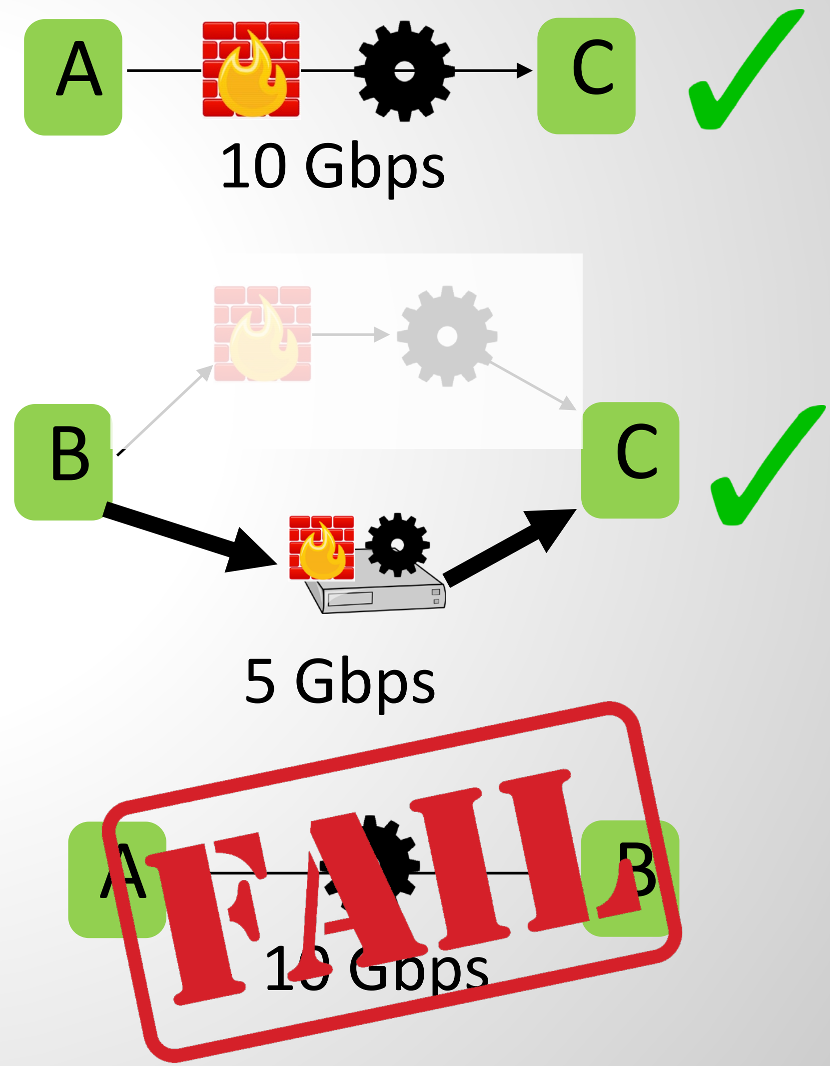
Which ones can be admitted and embedded?

Algorithmic Challenges: Admission Control and Embedding

Substrate:



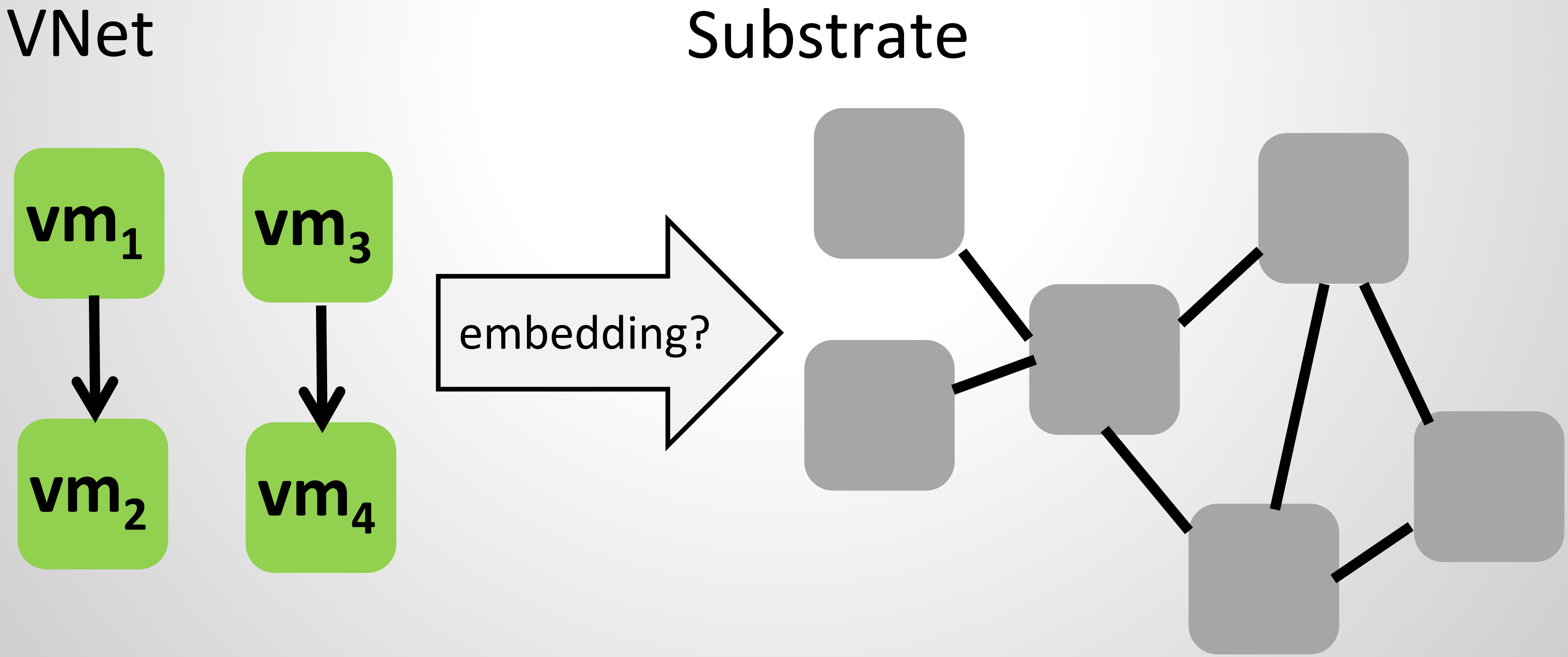
Requests:



Essentially a virtual network embedding problem!

The Virtual Network Embedding Problem

- A fundamental resource allocation problem
- 2 dimensions of flexibility:
  - Mapping of virtual nodes (to physical nodes)
  - Mapping of virtual links (to paths)



The Virtual Network Embedding Problem

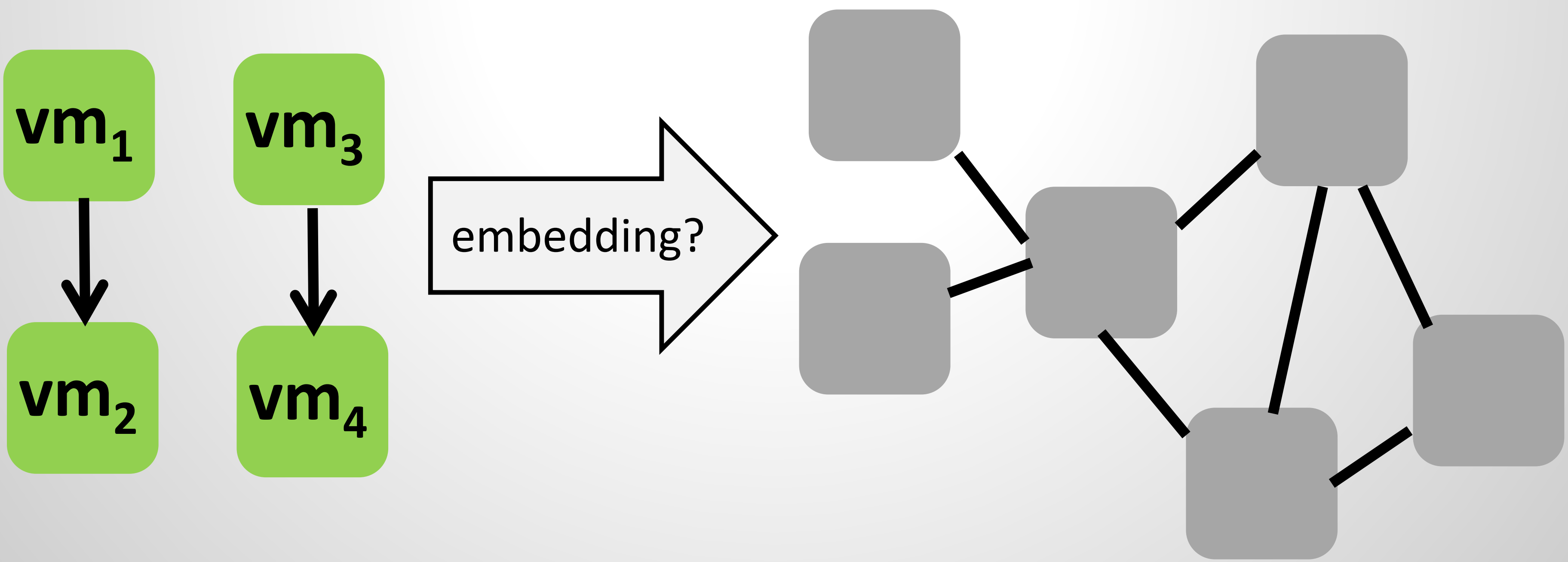
- A fundamental resource allocation problem
- 2 dimensions of flexibility:
  - ...des (to phy...
  - ...ks (to pat...

aka “guest graph”

aka “host graph”

VNet

Substrate

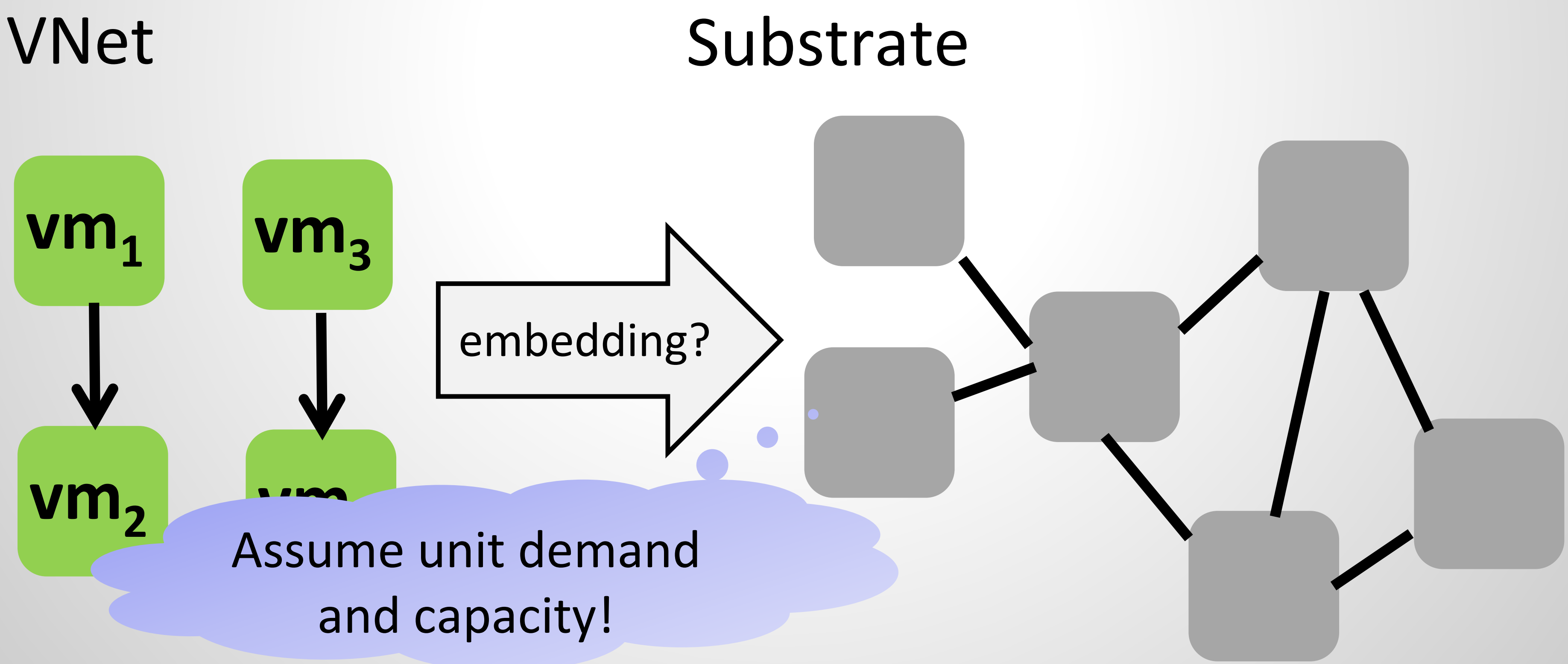


The Virtual Network Embedding Problem

- A fundamental resource allocation problem
- 2 dimensions of flexibility:
  - des (to phys
  - ks (to pat

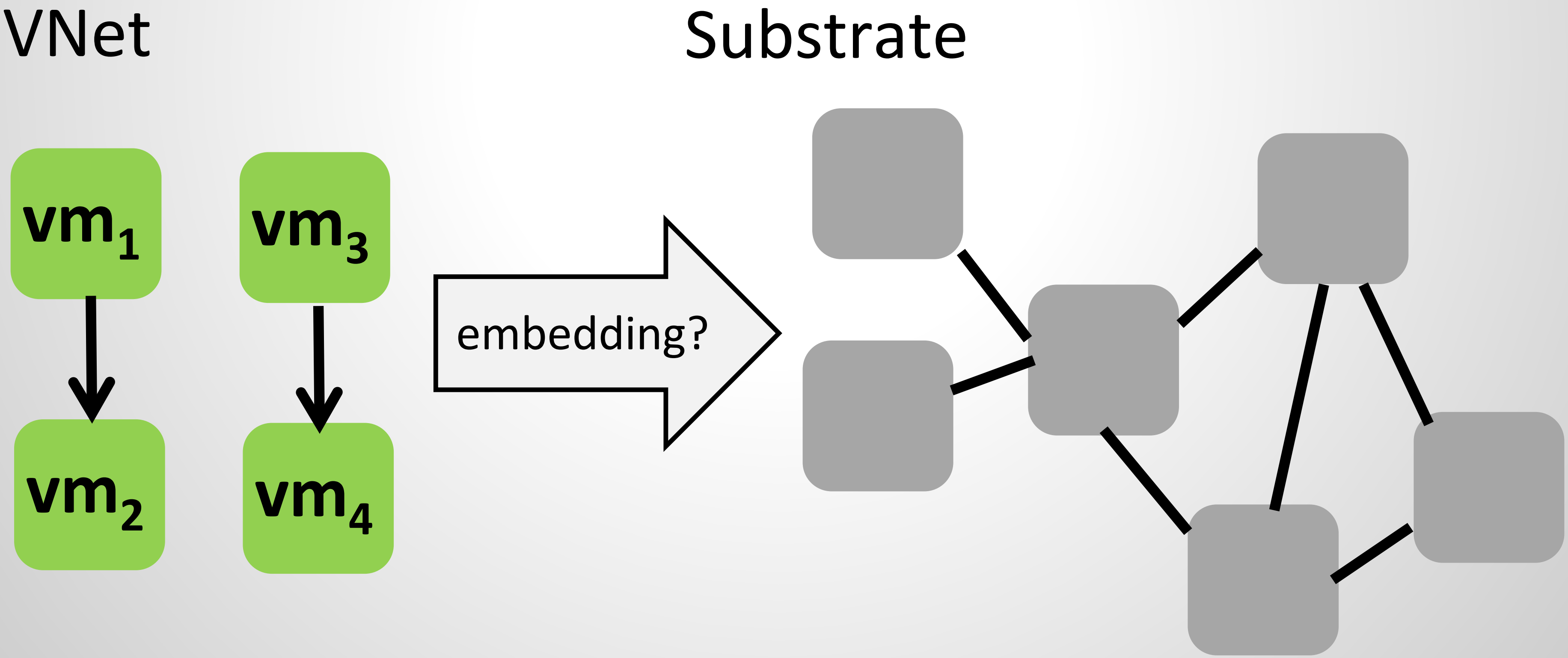
aka “guest graph”

aka “host graph”



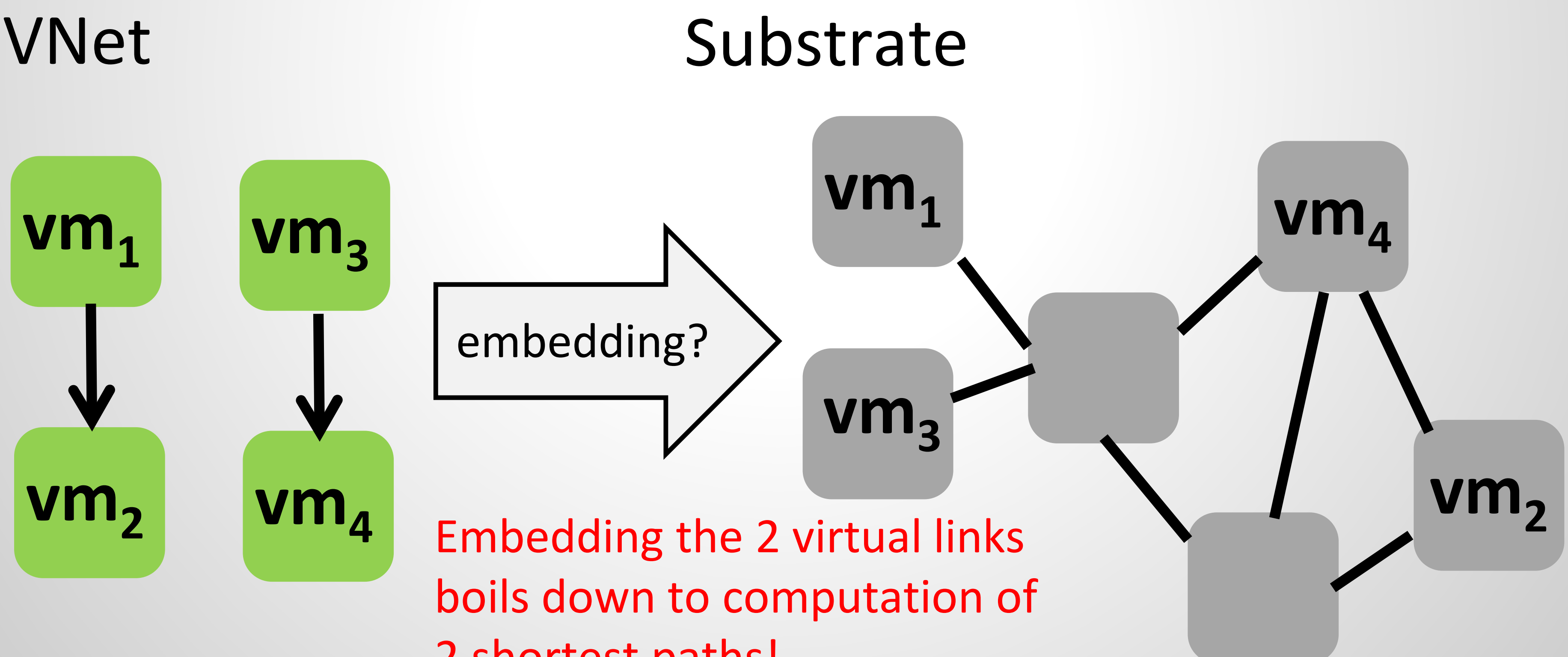
The Virtual Network Embedding Problem

- Let's start simple: assume node mappings are given



The Virtual Network Embedding Problem

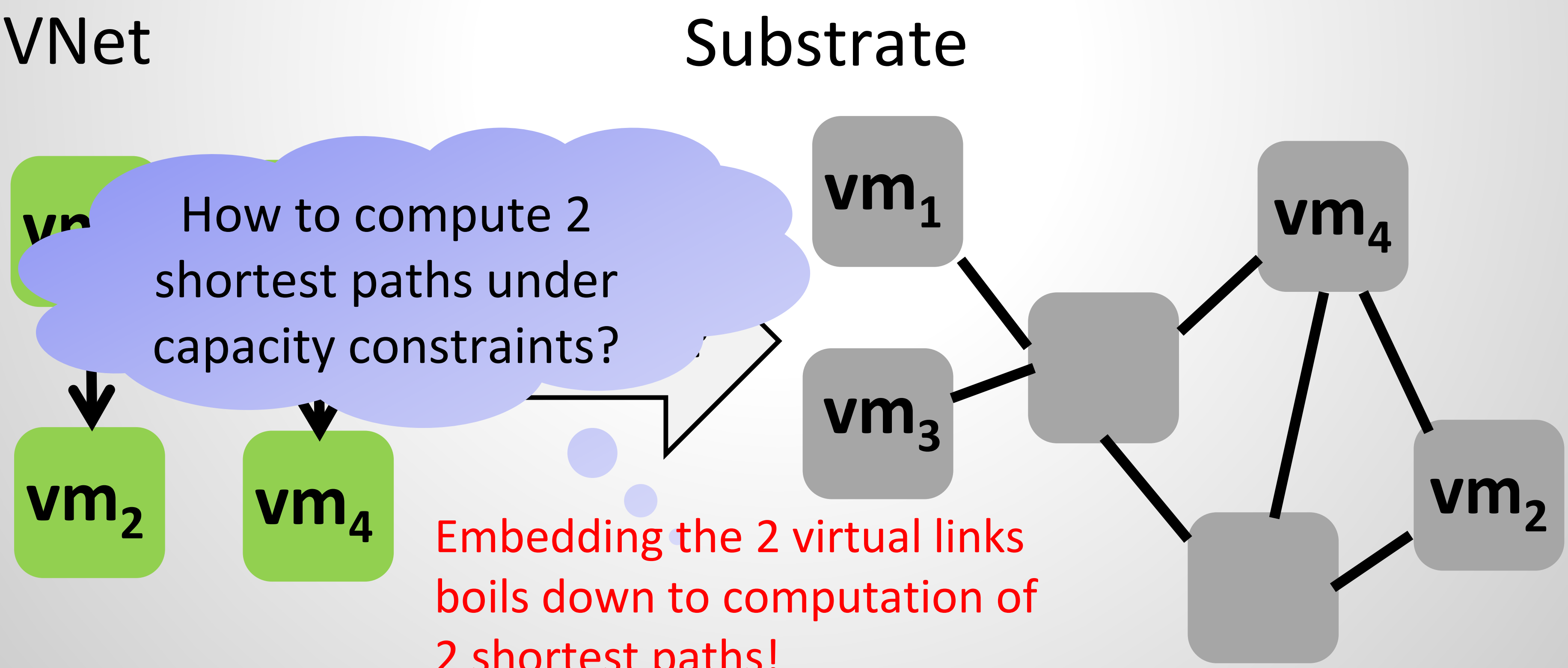
- Let’s start simple: assume node mappings are given



Embedding the 2 virtual links boils down to computation of 2 shortest paths!

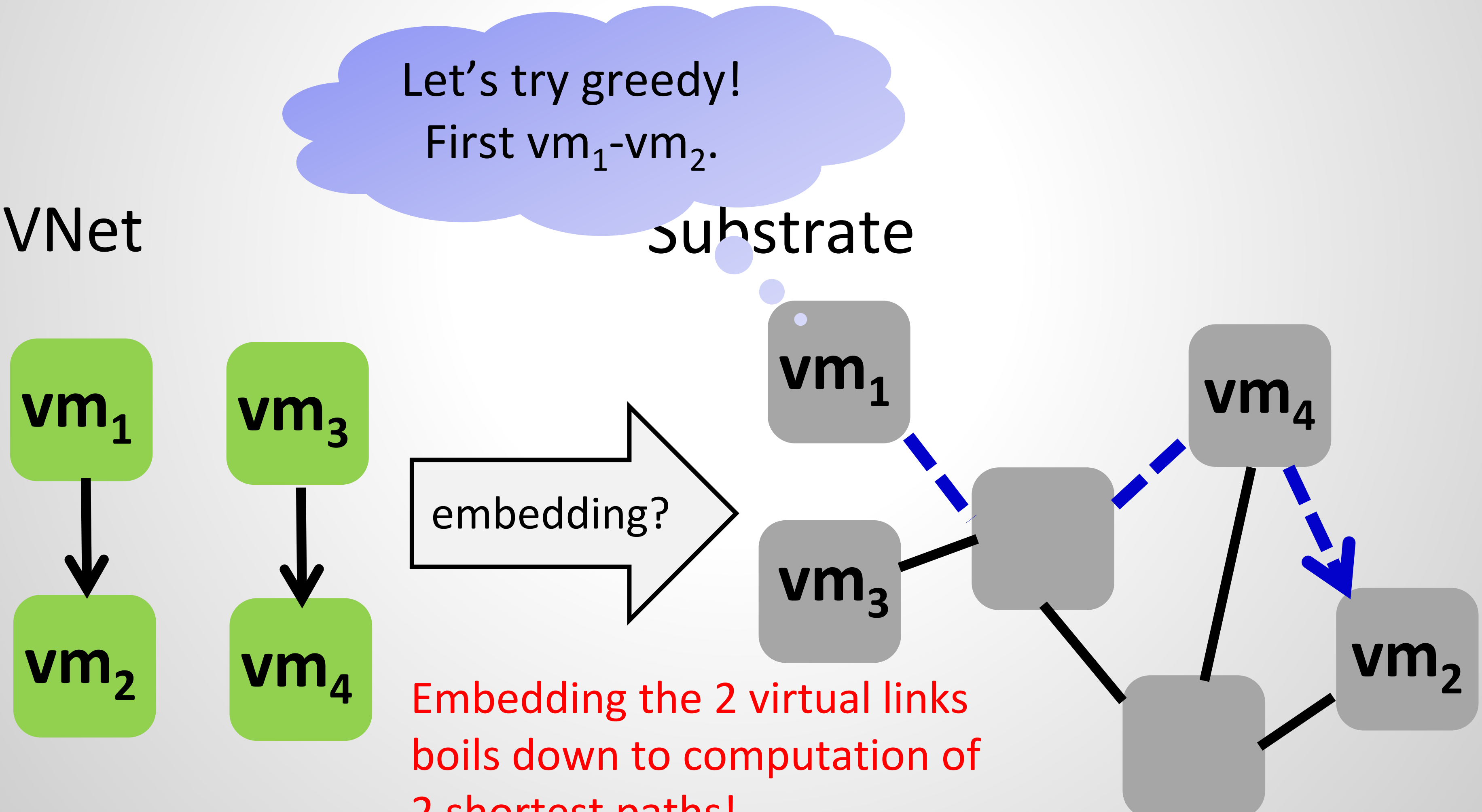
The Virtual Network Embedding Problem

- Let's start simple: assume node mappings are given



The Virtual Network Embedding Problem

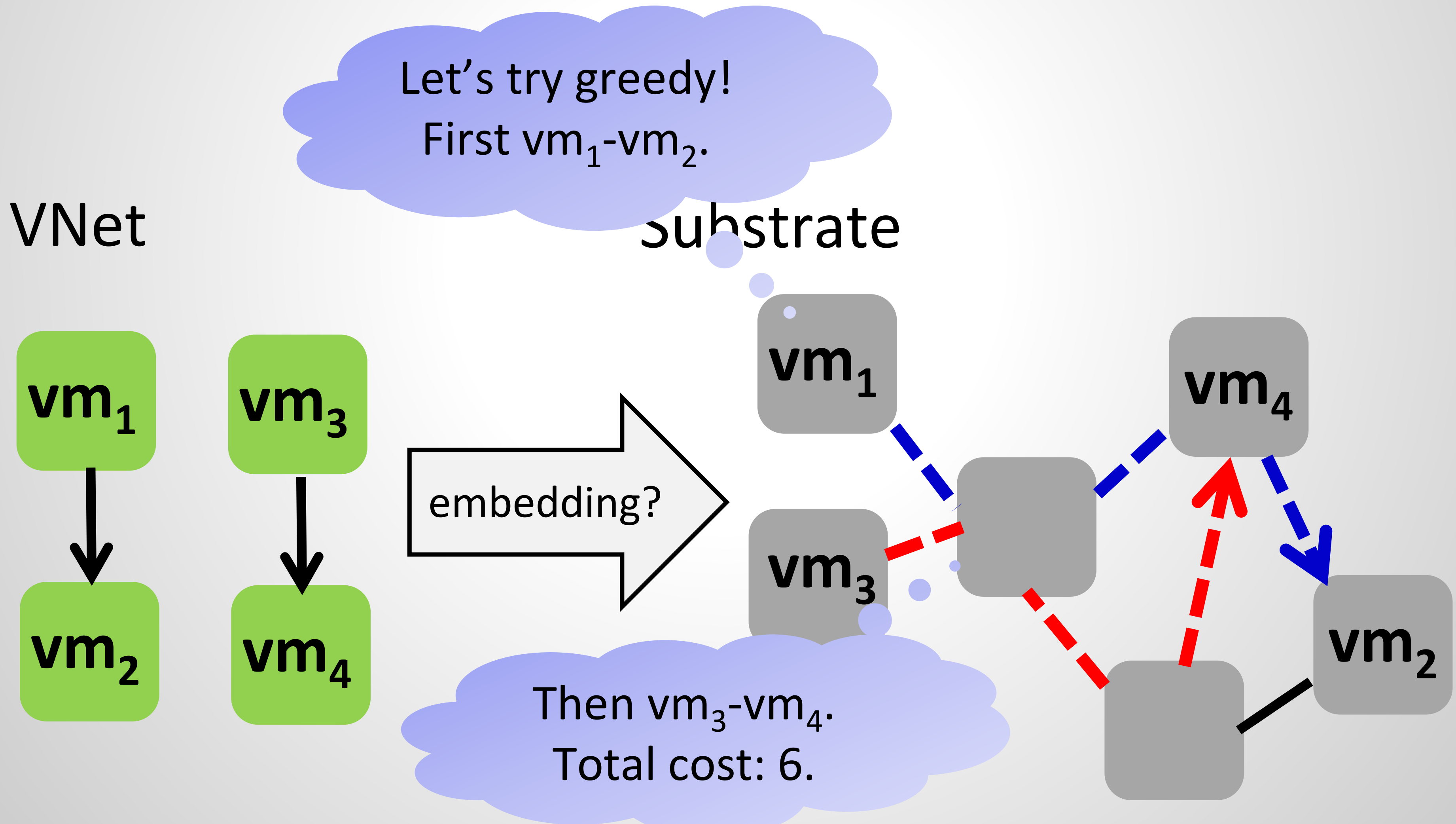
- Let's start simple: assume node mappings are given



Embedding the 2 virtual links boils down to computation of 2 shortest paths!

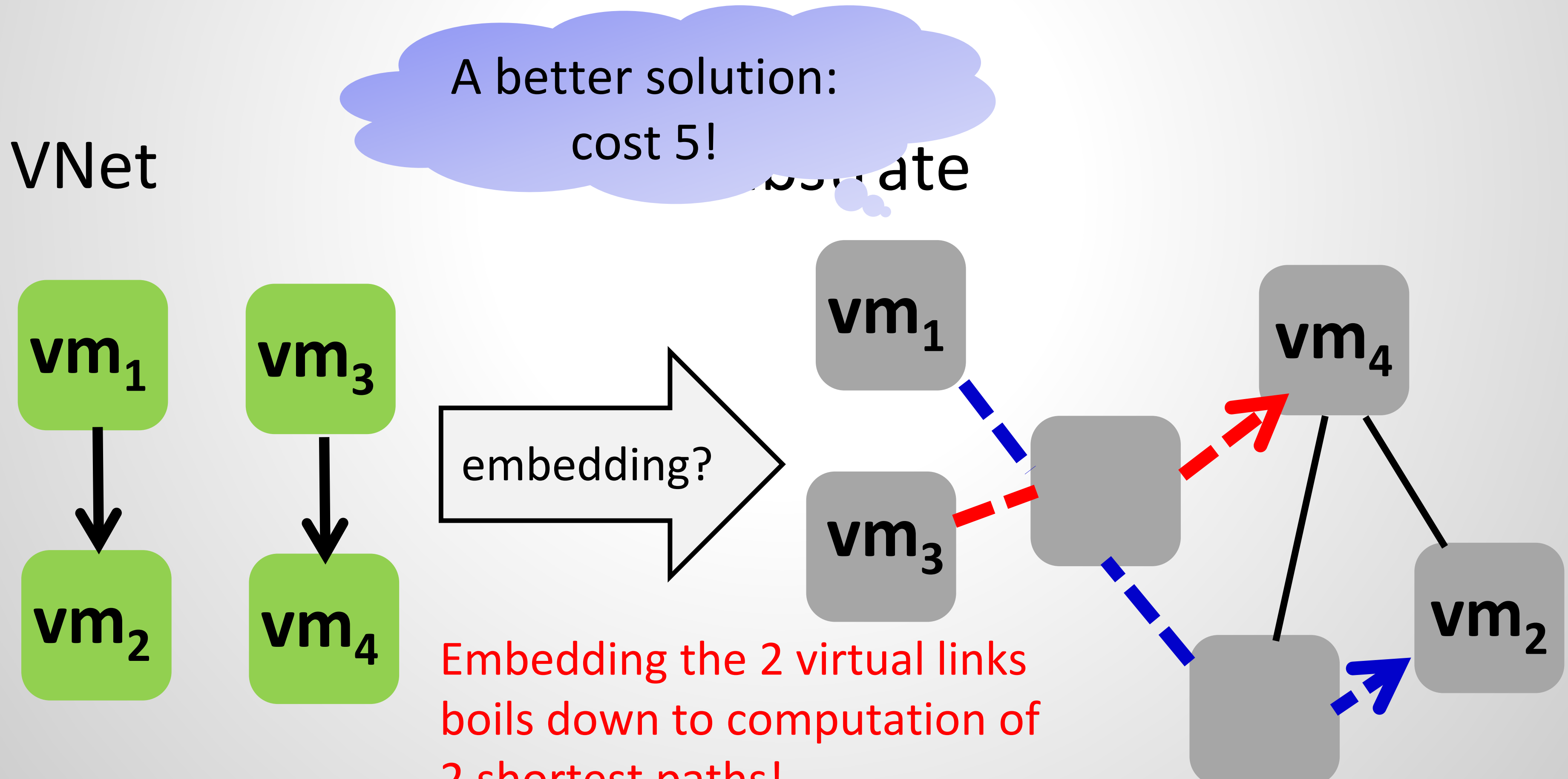
The Virtual Network Embedding Problem

- Let's start simple: assume node mappings are given



The Virtual Network Embedding Problem

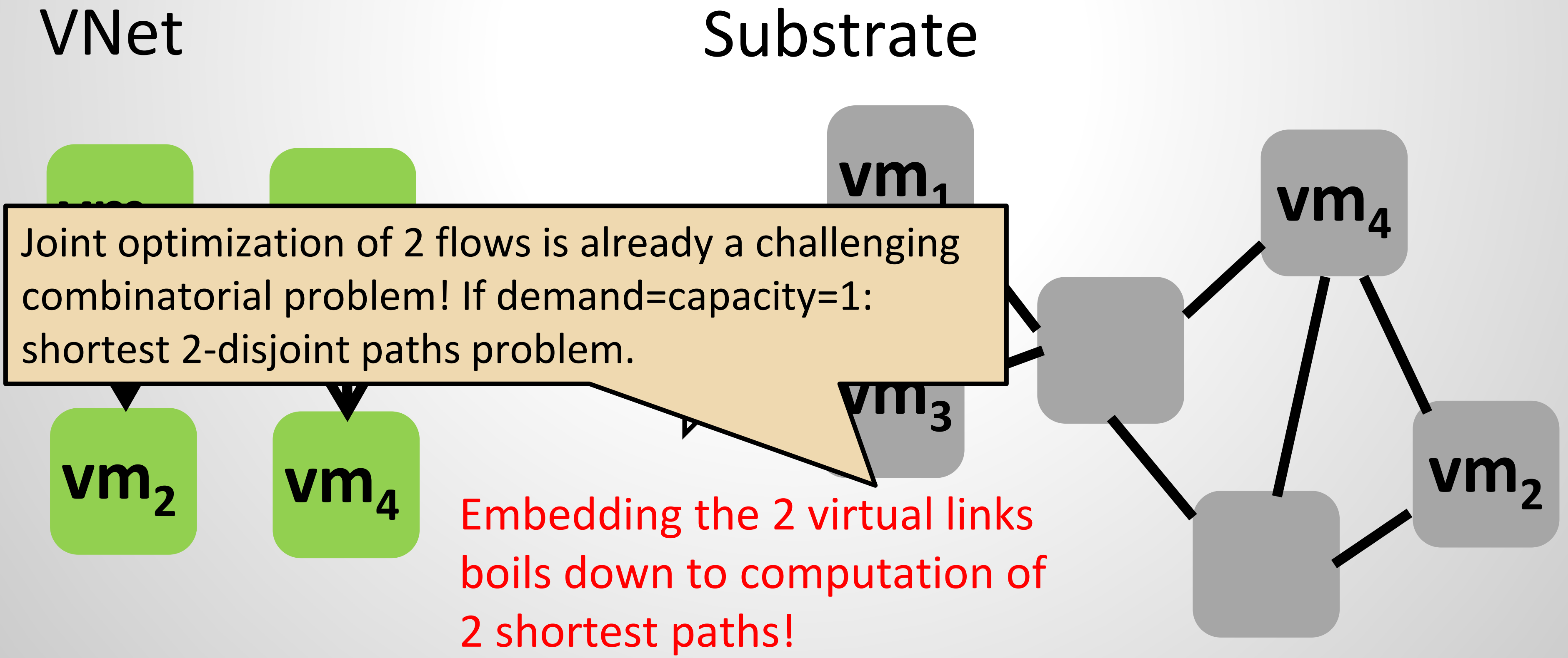
- Let's start simple: assume node mappings are given



Embedding the 2 virtual links boils down to computation of 2 shortest paths!

The Virtual Network Embedding Problem

- Let's start simple: assume node mappings are given

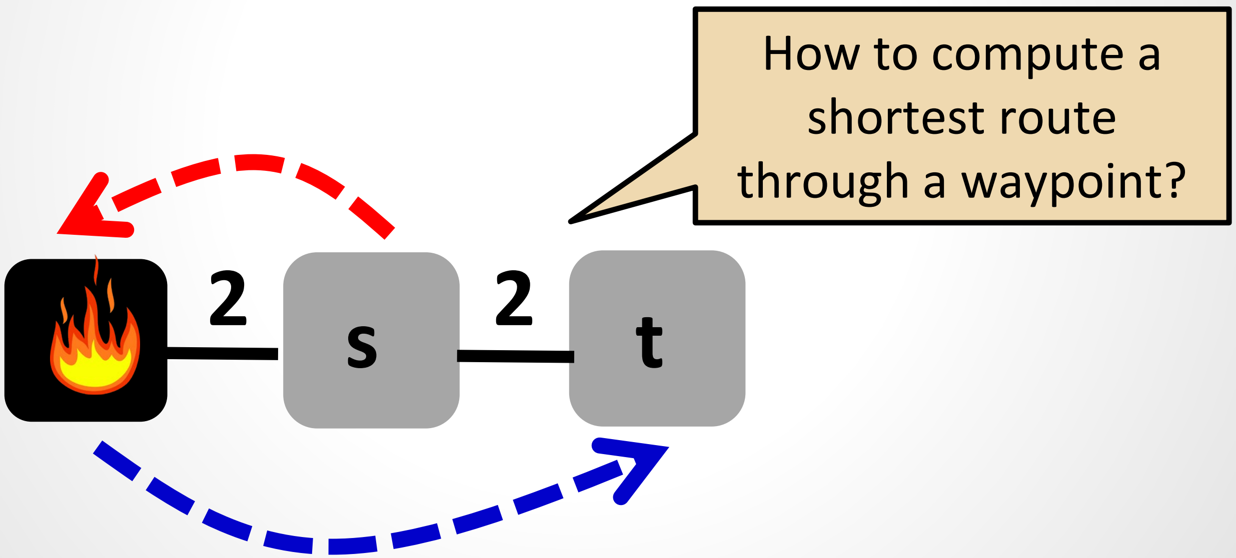


Therefore: Mapping Virtual Links is Challenging

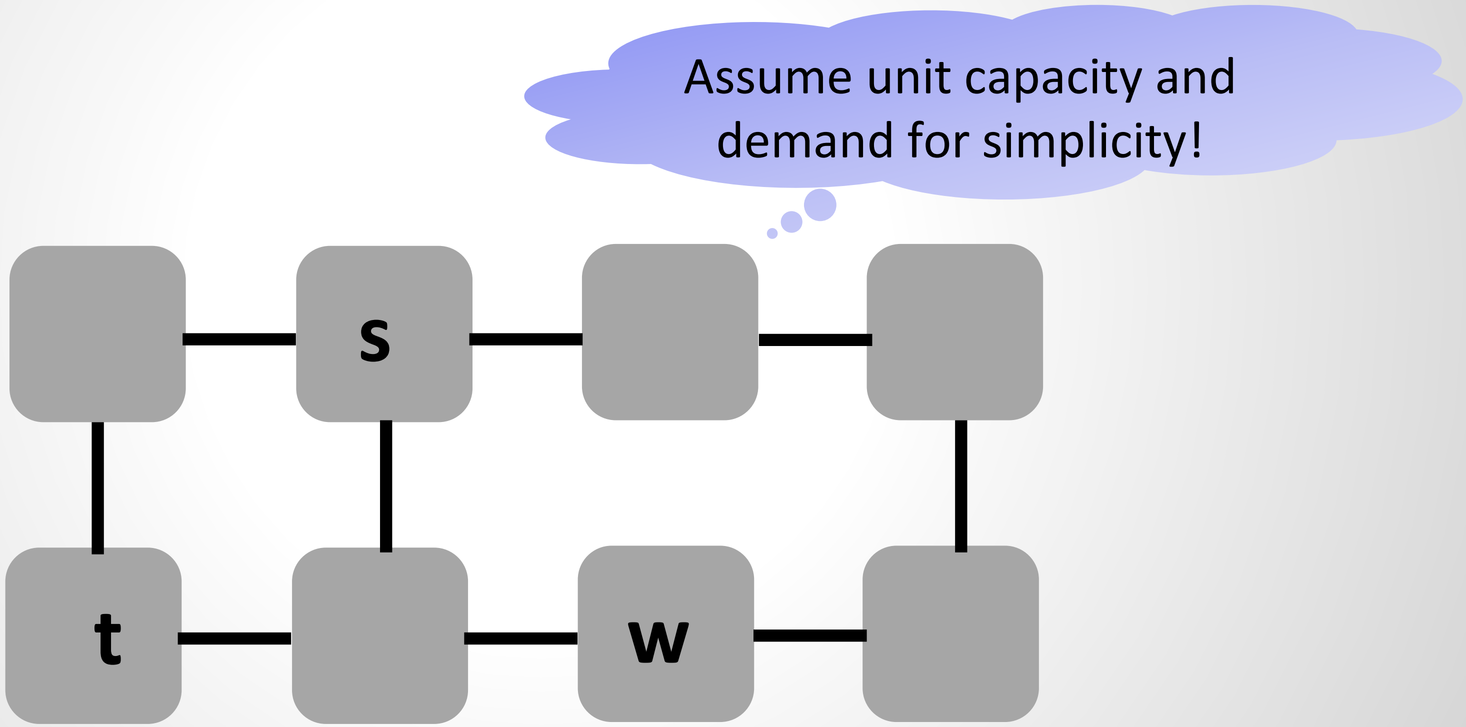
Bad news: The Virtual Network Embedding Problem is hard even if endpoints are already mapped and given.

Remark: Also Hard to Route 1 Waypoint!

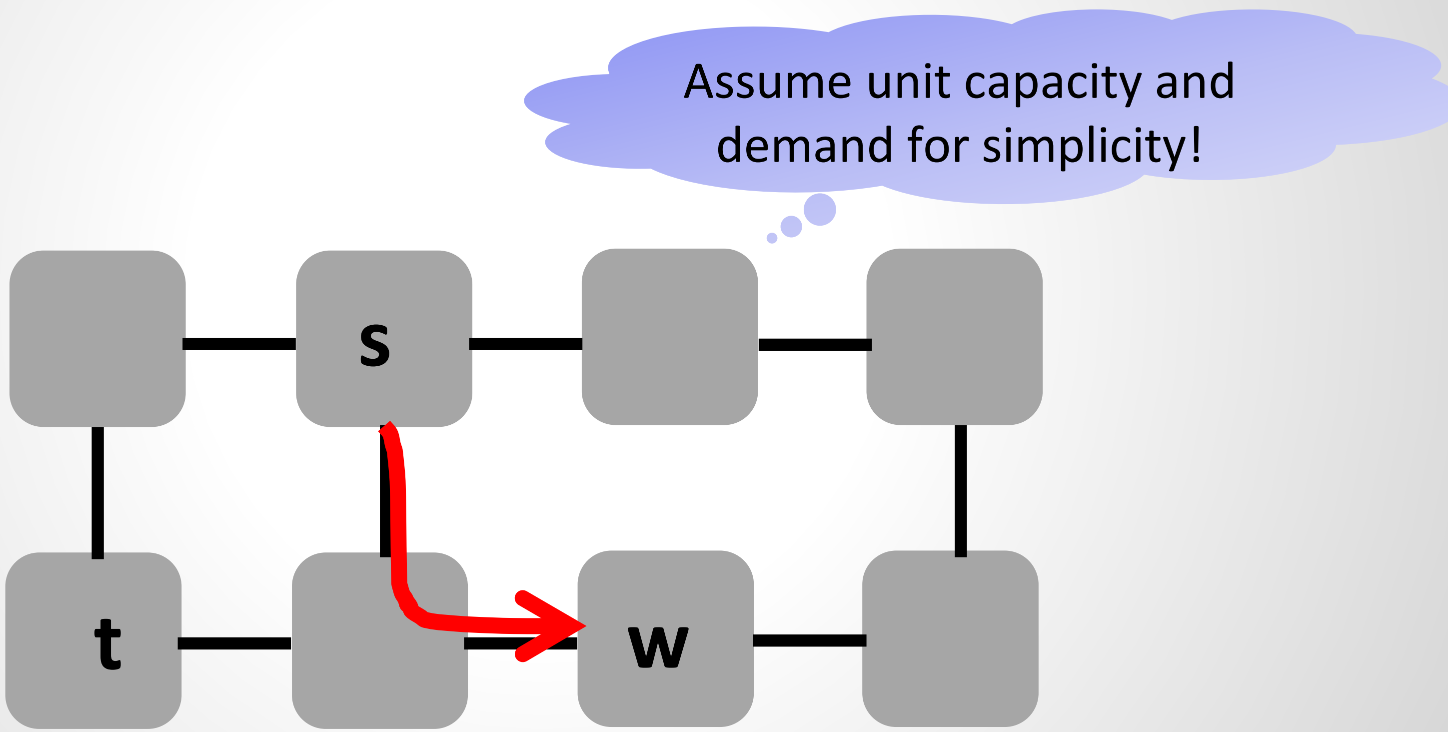
Steering traffic through a single network function / middlebox: a walk



Comuting A Shortest Walk Through A *Single* Given Waypoint is Non-Trivial!



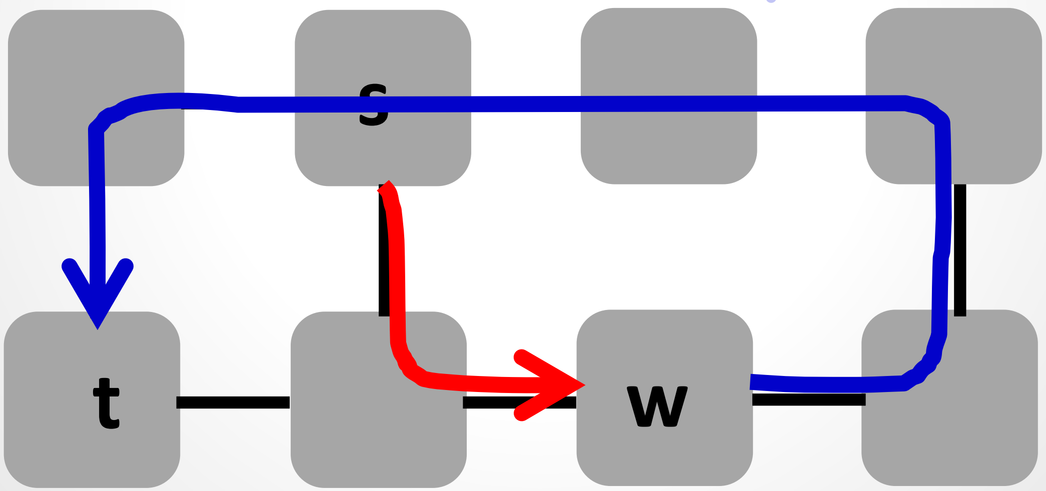
Comuting A Shortest Walk Through A *Single* Given Waypoint is Non-Trivial!



Greedy fails: **choose shortest path from s to w…**

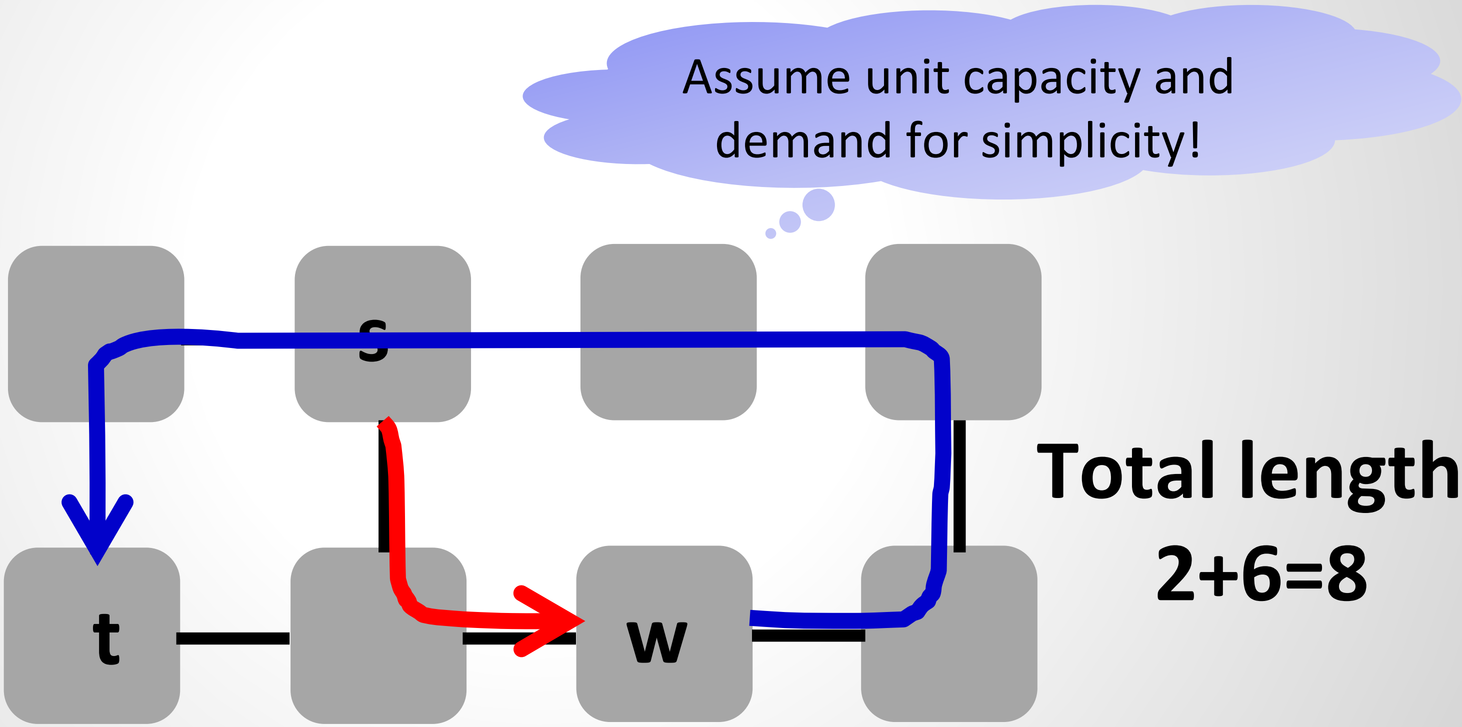
Comuting A Shortest Walk Through A *Single* Given Waypoint is Non-Trivial!

Assume unit capacity and demand for simplicity!



Greedy fails: **… now need long path from w to t**

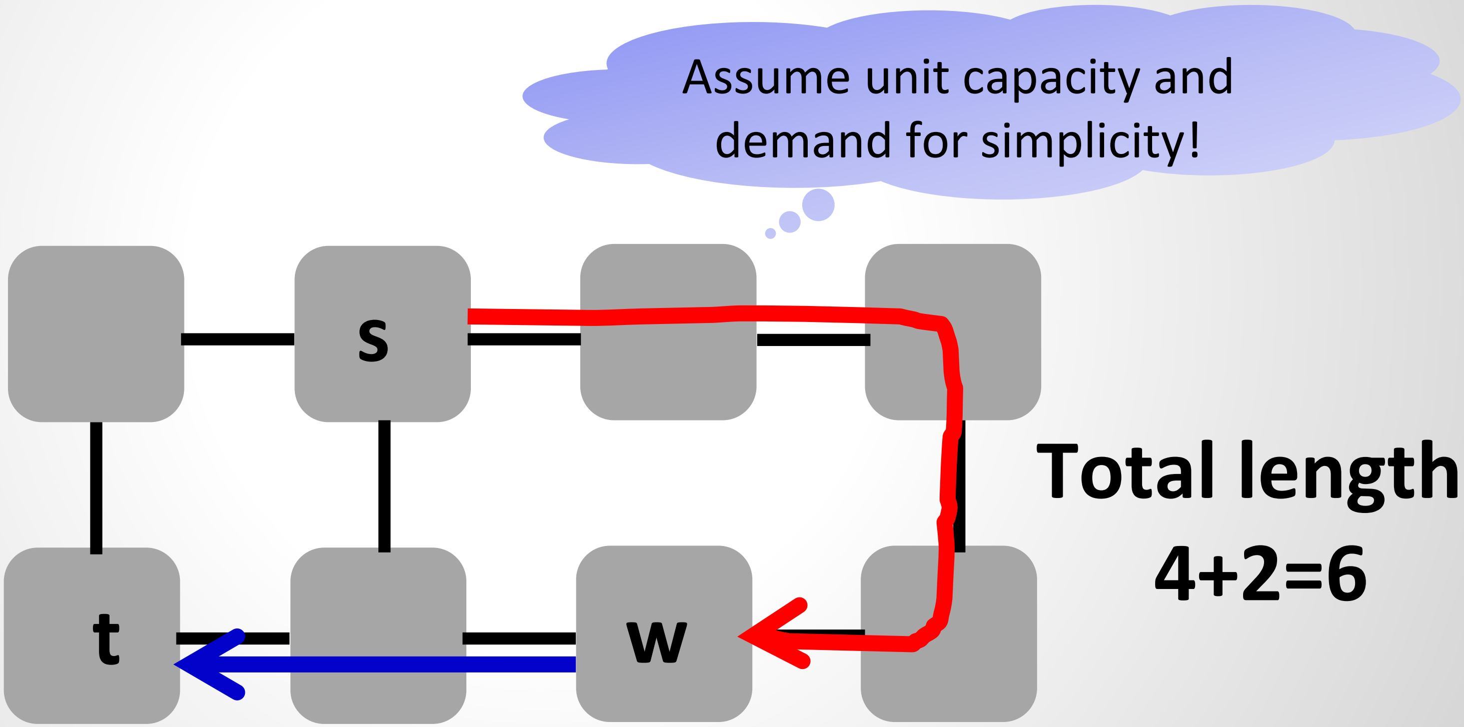
Comuting A Shortest Walk Through A *Single* Given Waypoint is Non-Trivial!



Total length: 2+6=8

Greedy fails: **… now need long path from w to t**

Comuting A Shortest Walk Through A *Single* Given Waypoint is Non-Trivial!

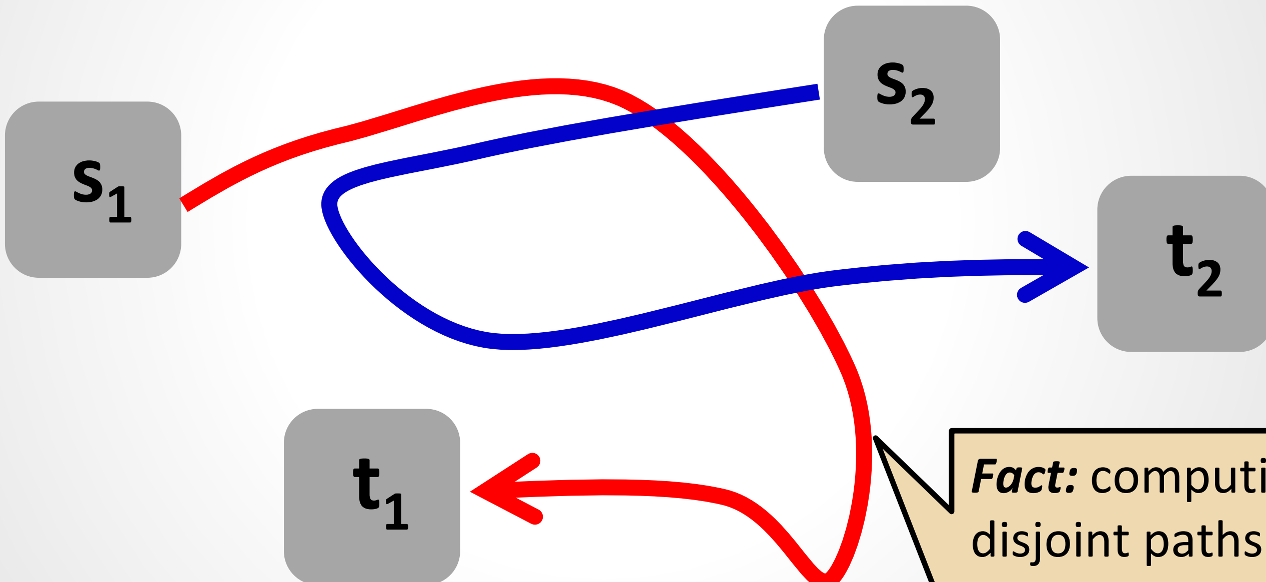


Total length: 4+2=6

A *better solution*: *jointly* optimize the two segments!

NP-hard on *Directed* Networks: Reduction from Disjoint Paths Problem

Reduction: *From* joint shortest paths $(s_1,t_1),(s_2,t_2)$ *to* shortest walk (s,w,t) problem

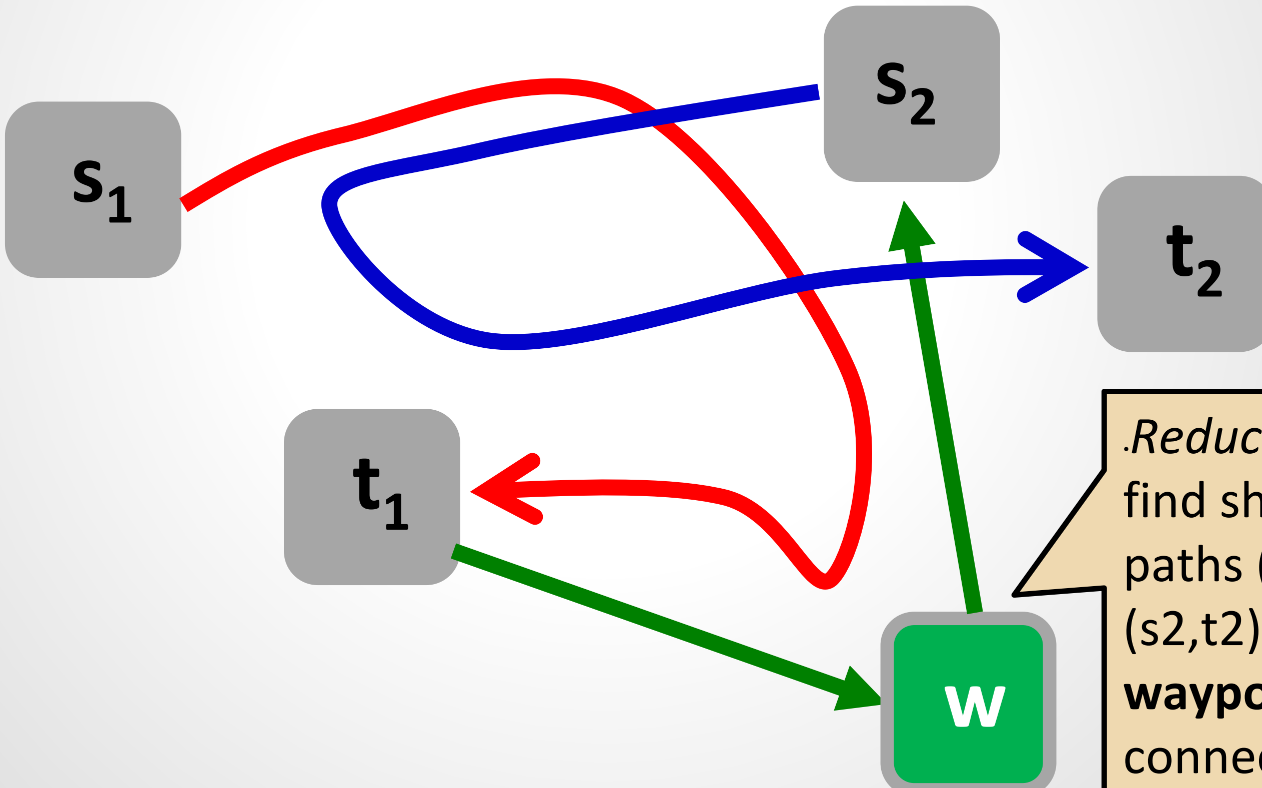


Fact: computing 2-disjoint paths (2DP) is NP-hard on directed graphs.

We show: If waypoint routing was in P, we could solve 2DP fast. ***Contradiction!***

NP-hard on *Directed* Networks: Reduction from Disjoint Paths Problem

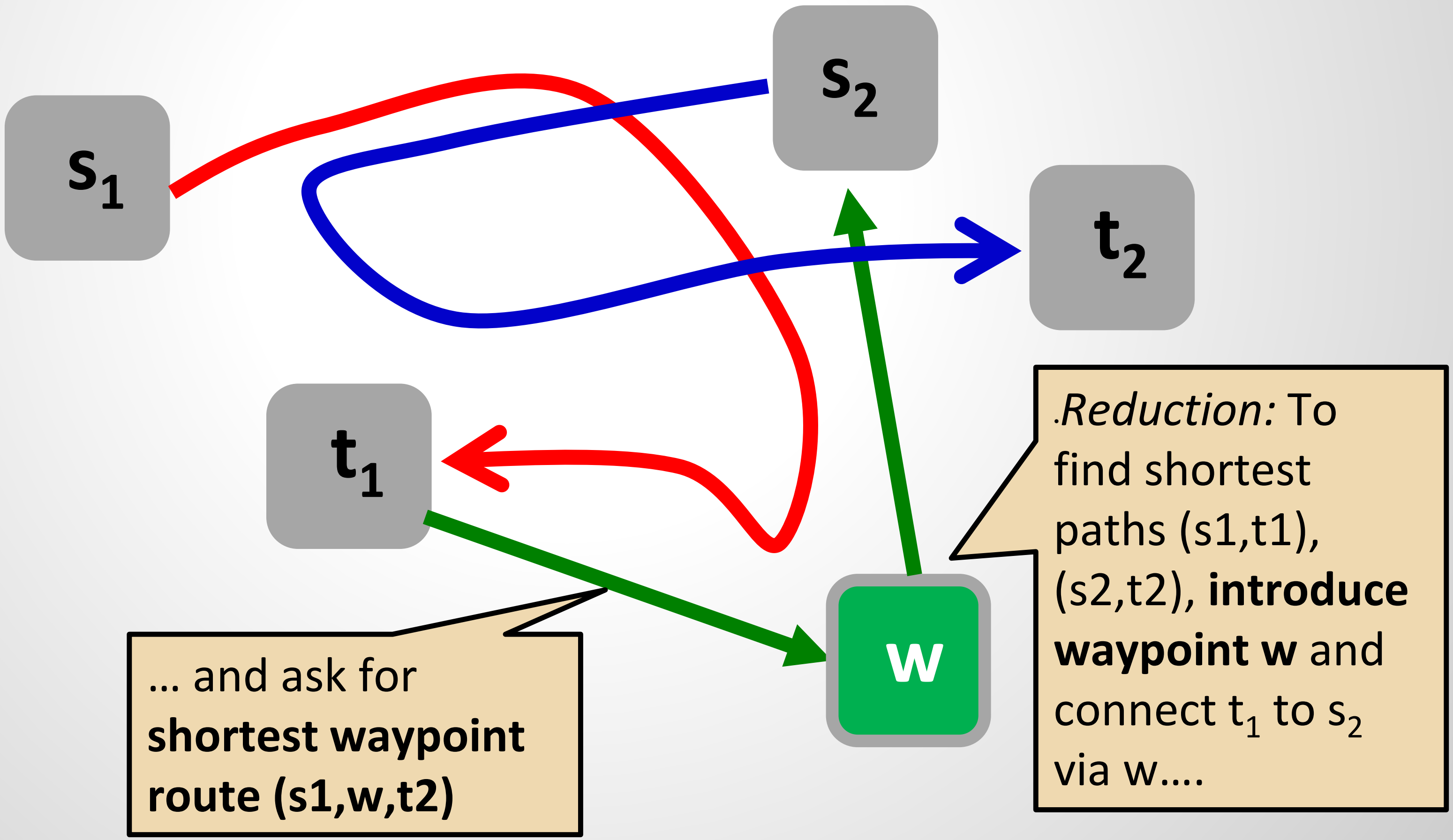
Reduction: *From* joint shortest paths $(s_1,t_1),(s_2,t_2)$ *to* shortest walk (s,w,t) problem



.*Reduction:* To find shortest paths (s1,t1), (s2,t2), **introduce waypoint w** and connect t_1 to s_2 via w….

NP-hard on *Directed* Networks: Reduction from Disjoint Paths Problem

Reduction: *From* joint shortest paths $(s_1,t_1),(s_2,t_2)$ *to* shortest walk (s,w,t) problem



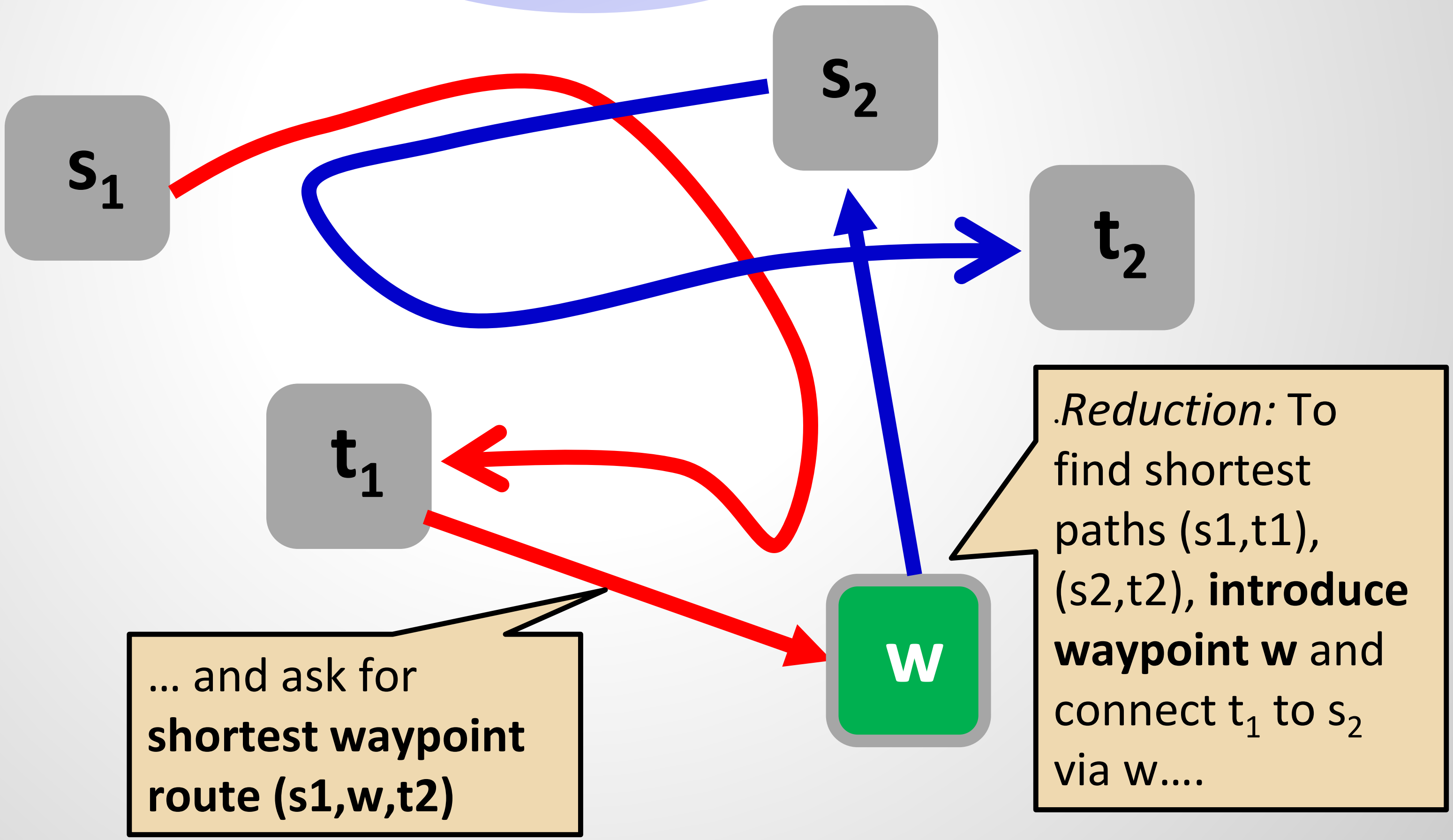
NP-hard on Direct… Netw…

F…

Reduc…

to shortest w…

The walk (s_1,w,t_2) walk defines a (s_1,t_1) and a (s_2,t_2) path pair before/after the waypoint! Solves original problem: **Contradiction!**

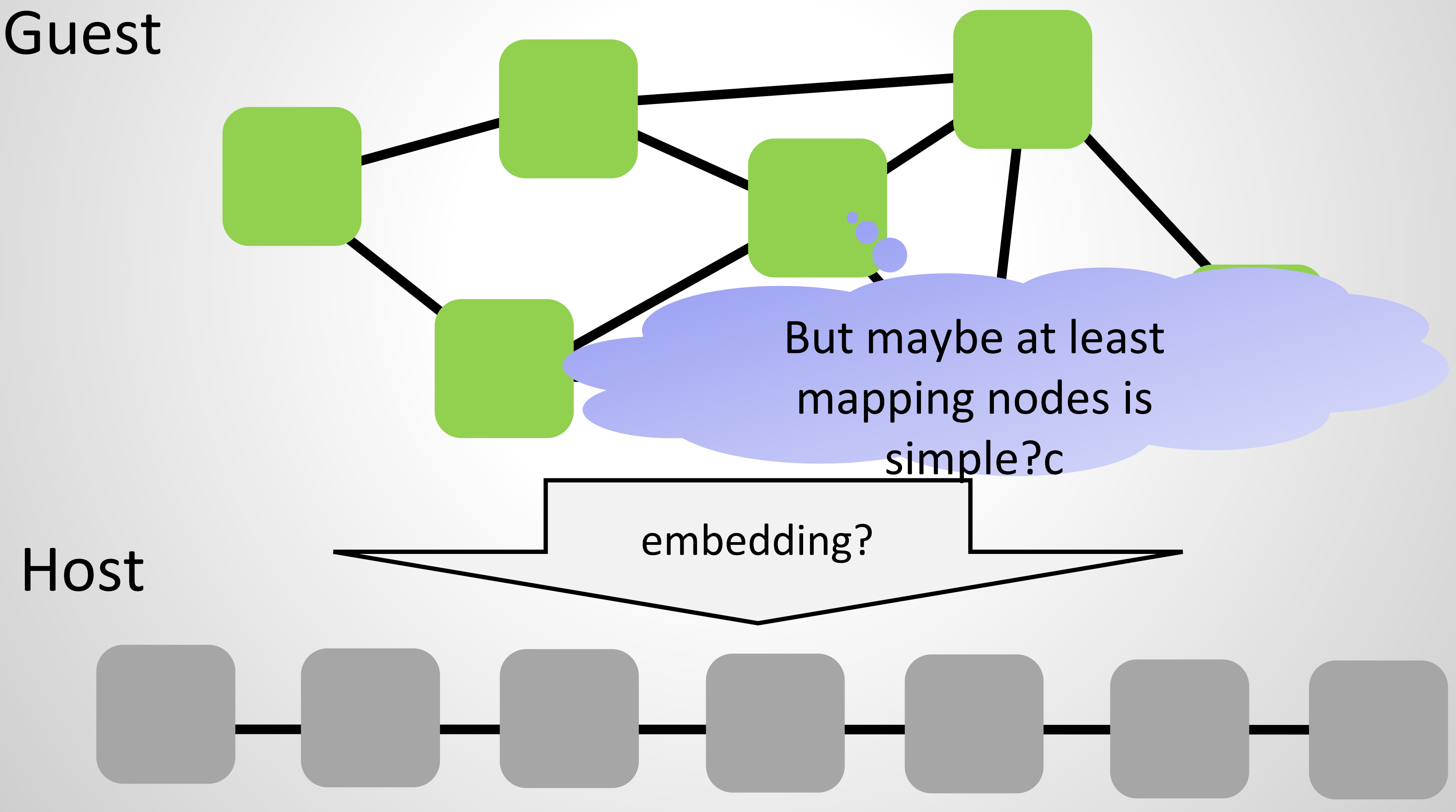


Mapping Virtual Nodes

Routing is hard! Maybe at least mapping nodes is simple?

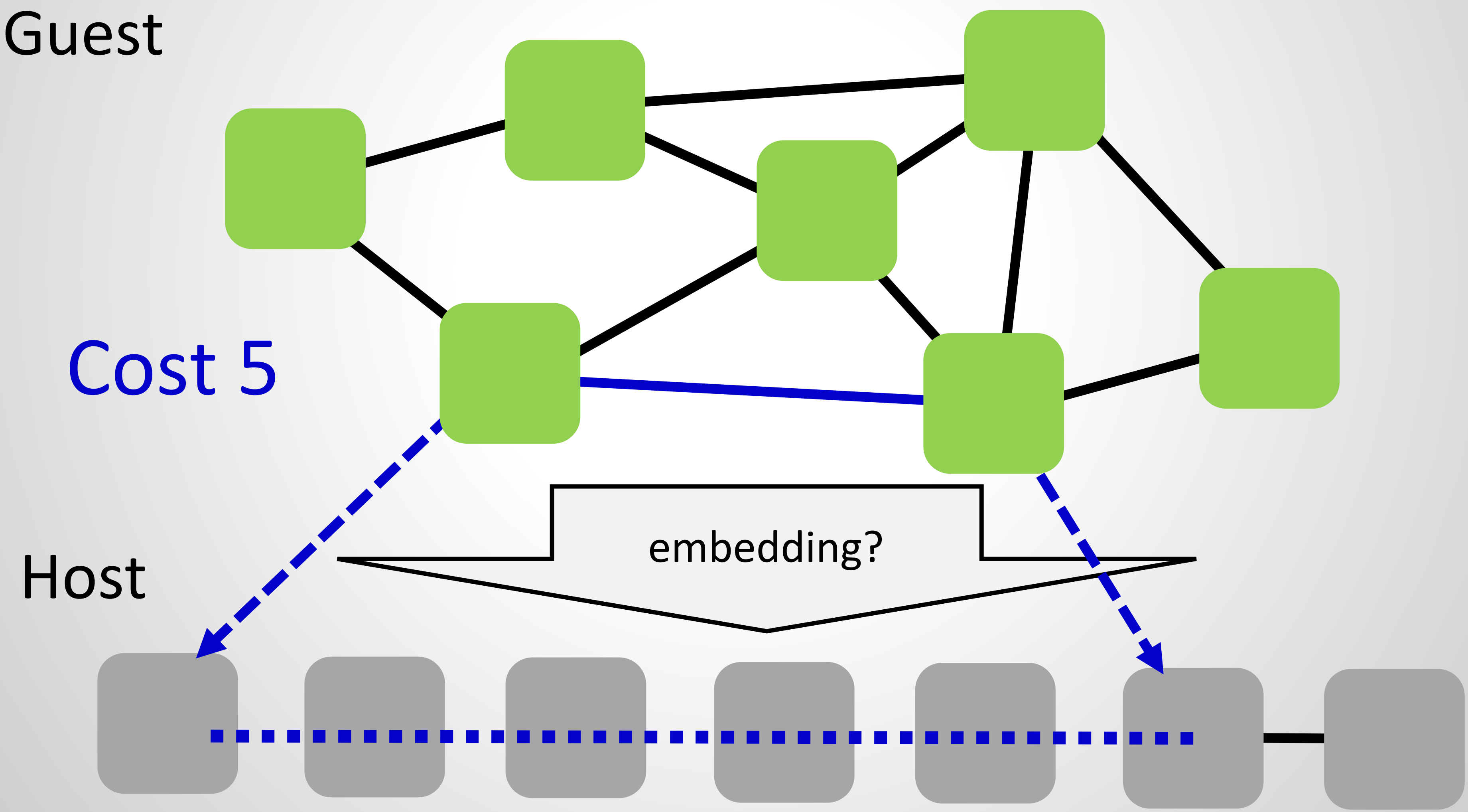
Mapping Virtual Nodes

- Let’s start simple again: assume paths are trivial, e.g., the physical network (host graph) is a line



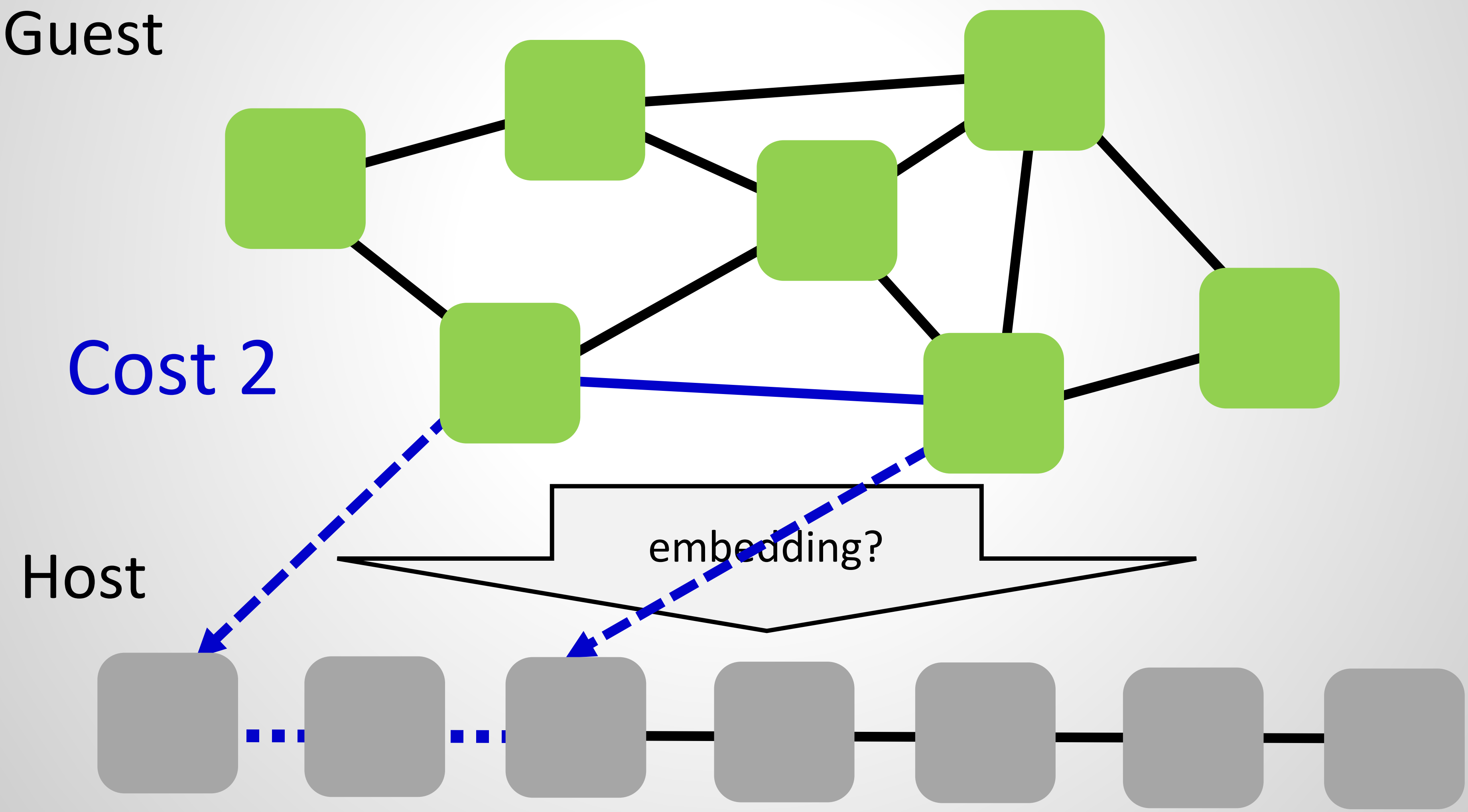
Mapping Virtual Nodes

- Let’s start simple again: assume paths are trivial, e.g., the physical network (host graph) is a line



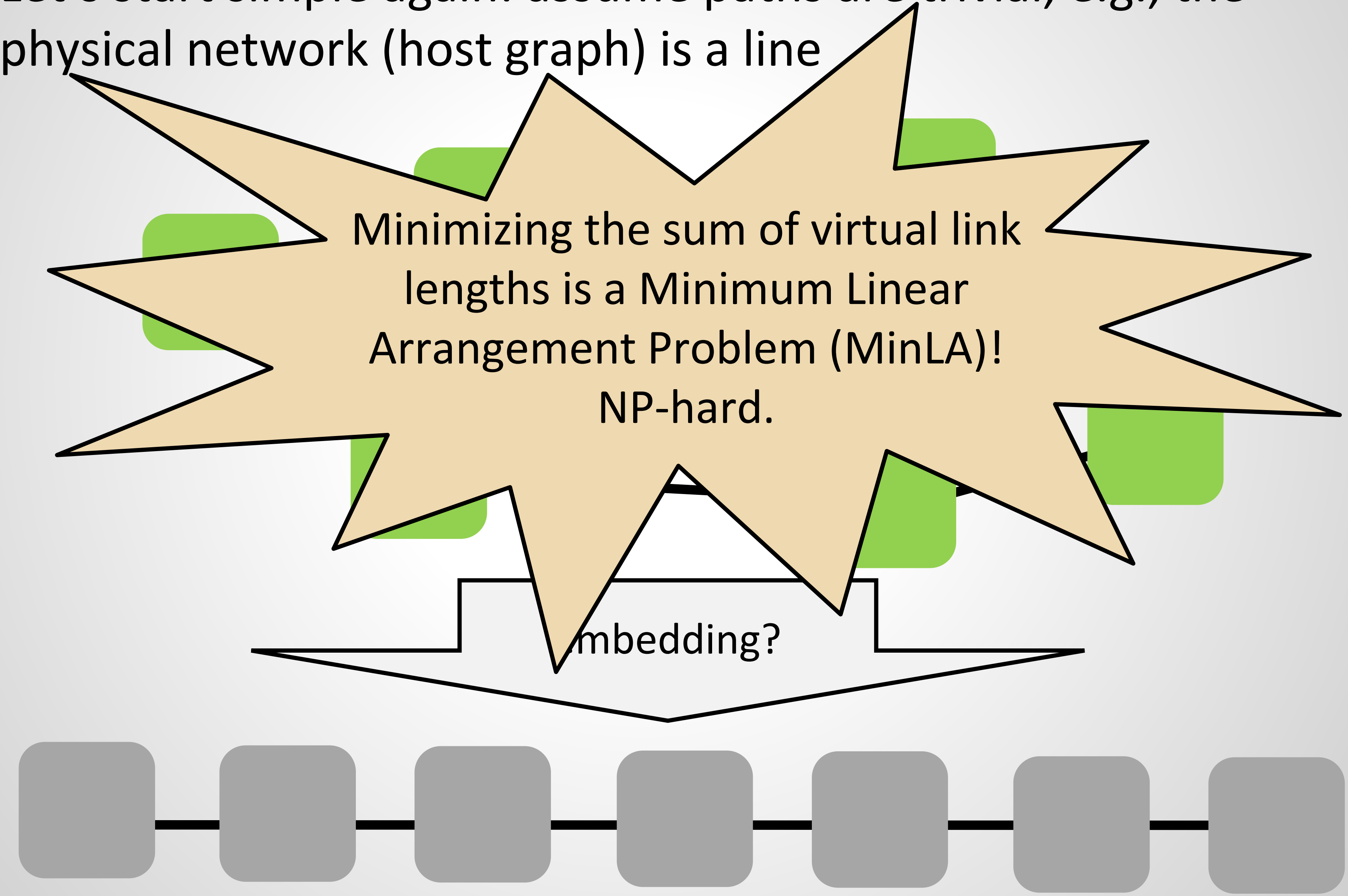
Mapping Virtual Nodes

- Let’s start simple again: assume paths are trivial, e.g., the physical network (host graph) is a line



Mapping Virtual Nodes

- Let’s start simple again: assume paths are trivial, e.g., the physical network (host graph) is a line



Therefore: VNEP is Hard "in Both Dimensions"!

- We have seen examples that:
  - mapping virtual links is hard (even if nodes are given)
  - mapping virtual nodes is hard (even if links are trivial)

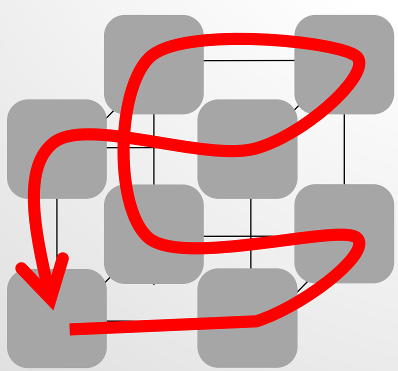
- Remark: the VNEP can also be seen as a generalization of the Subgraph Isomorphism Problem (SIP)

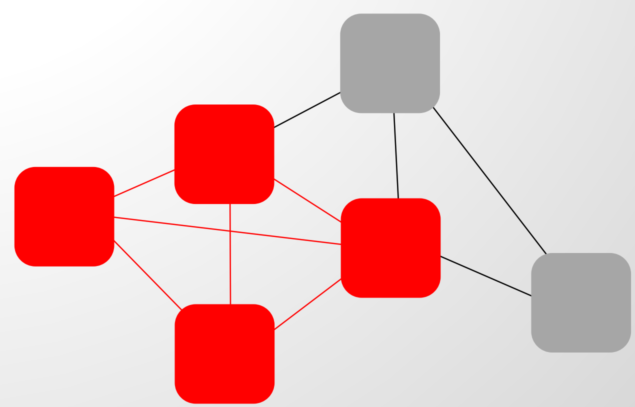
Known? Why is SIP NP-hard?

Therefore: VNEP is Hard "in Both Dimensions"!

- We have seen examples that:
  - mapping virtual links is hard (even if nodes are given)
  - mapping virtual nodes is hard (even if links are trivial)

- Remark: the VNEP can also be seen as a generalization of the Subgraph Isomorphism Problem (SIP)
  - The SIP problem: Given two graphs G,H, determine whether G contains a subgraph that is isomorphic to H?
  - NP-hard: "does G contain an n-node cycle?" is a Hamilton cycle problem (each node visited exactly once), a solution to "does G contain a k-clique?" solves maximum clique problem, etc.



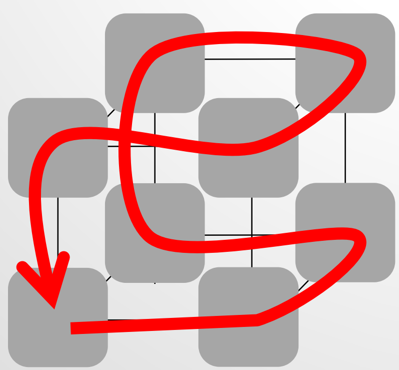


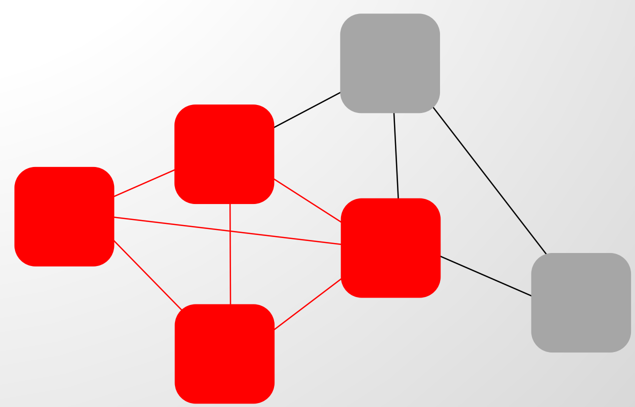
Therefore: VNEP is Hard “in Both Dimensions”!

- We have seen examples that:
  - mapping virtual links is hard (even if nodes are given)
  - mapping virtual nodes is hard (even if links are trivial)

- Remark: the VNEP can also be seen as a generalization of the Subgraph Isomorphism Problem (SIP)
  - The SIP problem: Given two graphs G,H, determine whether G contains a subgraph that is isomorphic to H?
  - N… …milton cycle problem (each node visited … ?” solves maximum clique problem, etc.

So if SIP is hard, why is VNEP hard?





NP-Hardness: From SIP to VNEP

- Observe: VNEP is a generalization of SIP

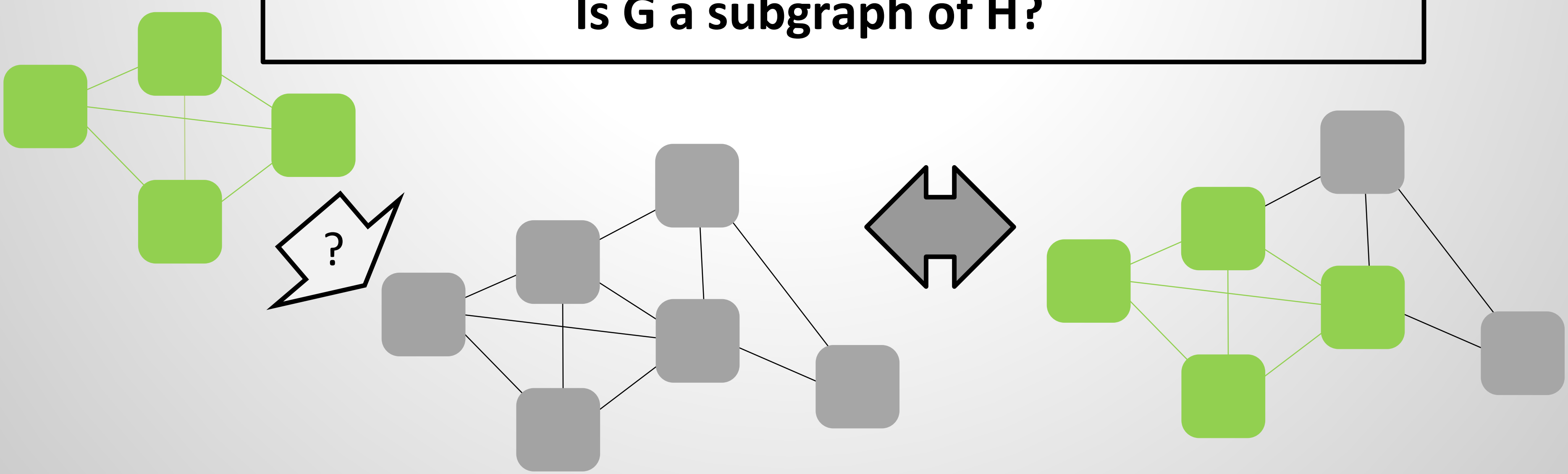
- For example:

Can VNet G=(V,E) be embedded in H at cost |E|?

(I.e., each virtual edge has length 1.)

⇔

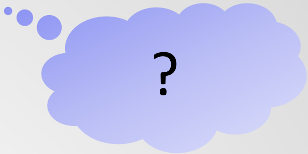
Is G a subgraph of H?



Can we at least formulate a “fast” MIP?

The basis for approximation algorithms
and heuristics! Even online!
E.g., relaxation and rounding.

Can we at least formulate a “fast” MIP?



Can we at least formulate a “fast” MIP?

- Recall: Mixed Integer Program (MIP)
  - Linear objective function (e.g., minimize embedding footprint)
  - Linear constraints (e.g., do not violate capacity constraints)
- Solved, e.g., with branch-and-bound search tree

Can we at least formulate a "fast" MIP?

- Recall: Mixed Integer Program (MIP
  - Linear objective function (e.g., minimize
  - Linear constraints (e.g., do not violate capacity constra

One that provides good relaxations!

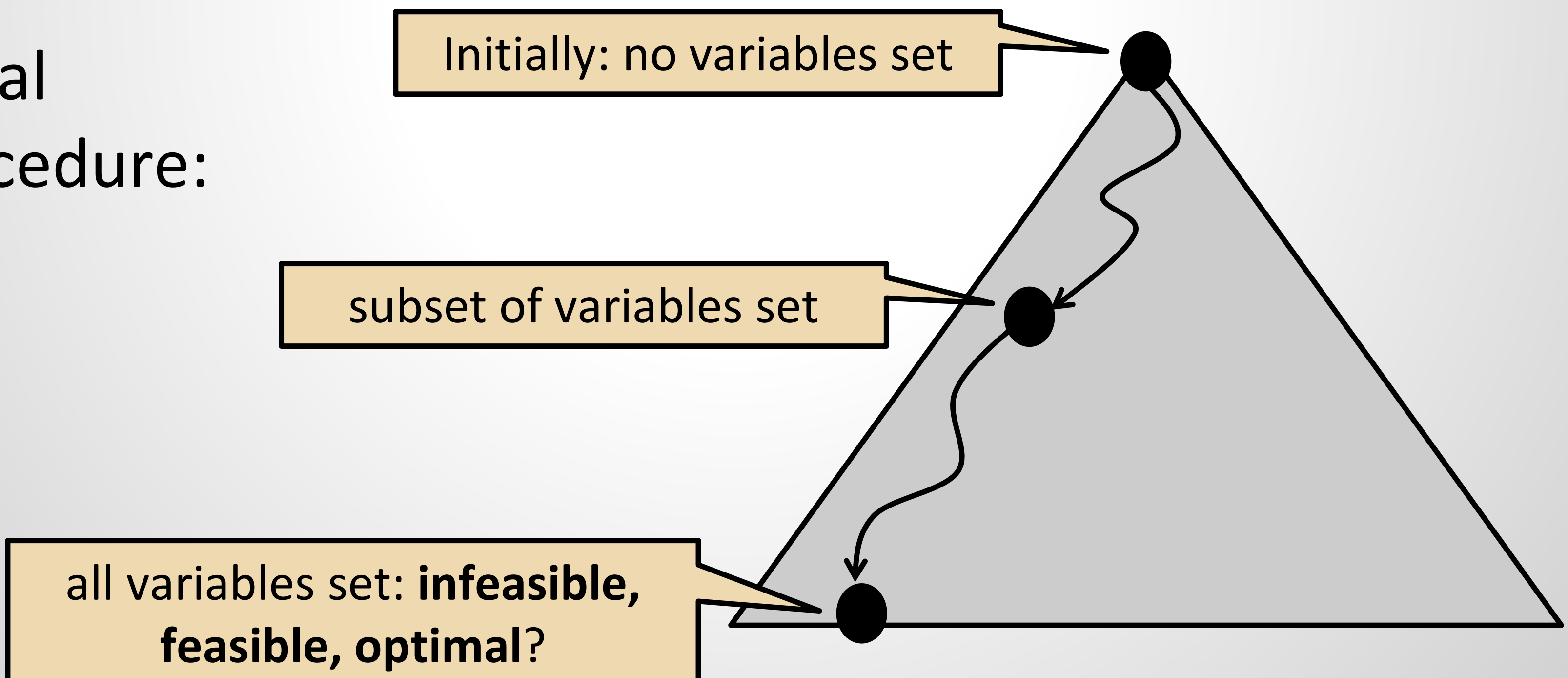
- Solved, e.g., with branch-and-bound search tree

Can we at least formulate a “fast” MIP?

- Recall: Mixed Integer Program (MIP)
  - Linear objective function (e.g., minimize embedding footprint)
  - Linear constraints (e.g., do not violate capacity constraints)

- Solved, e.g., with branch-and-bound search tree

Usual procedure:

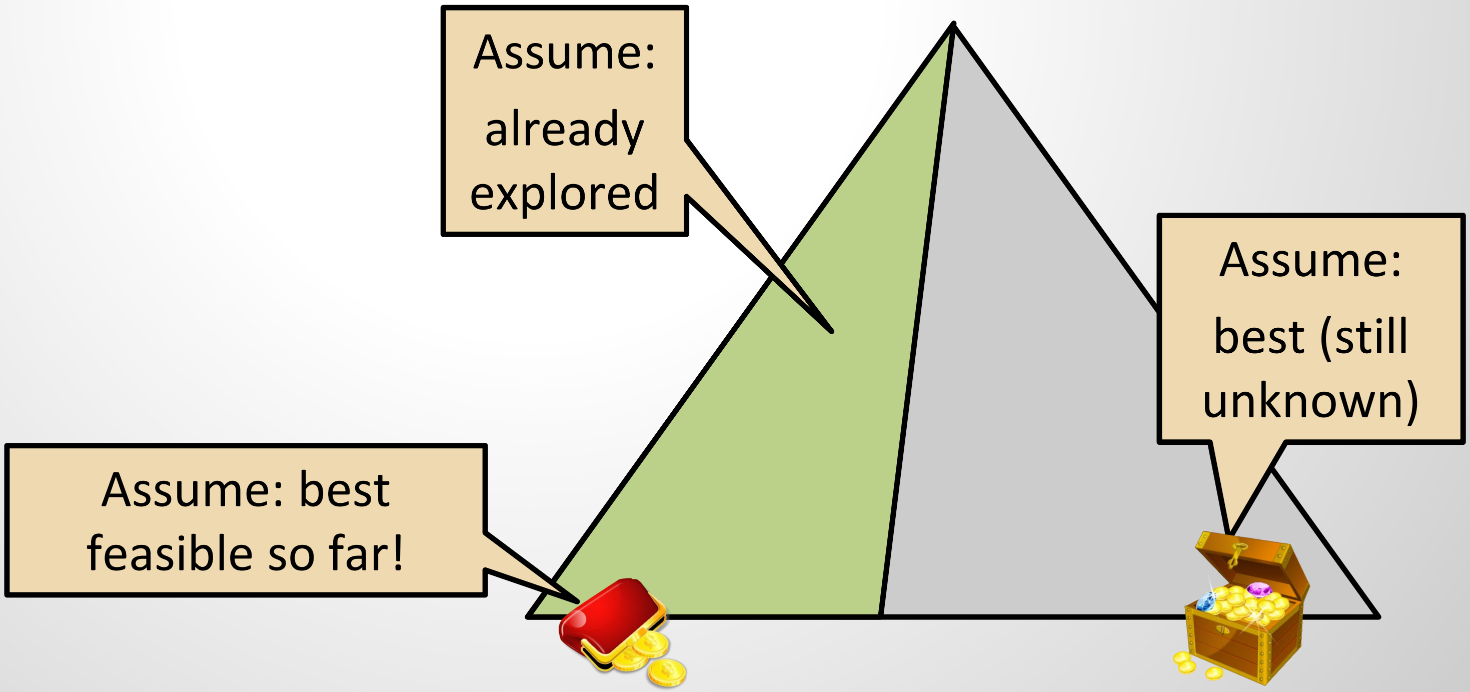


Can we at least formulate a “fast” MIP?

- Recall: Mixed Integer Program (MIP)
  - Linear objective function (e.g., minimize embedding footprint)
  - Linear constraints (e.g., do not violate capacity constraints)

- Solved, e.g., with branch-and-bound search tree

Usual procedure:

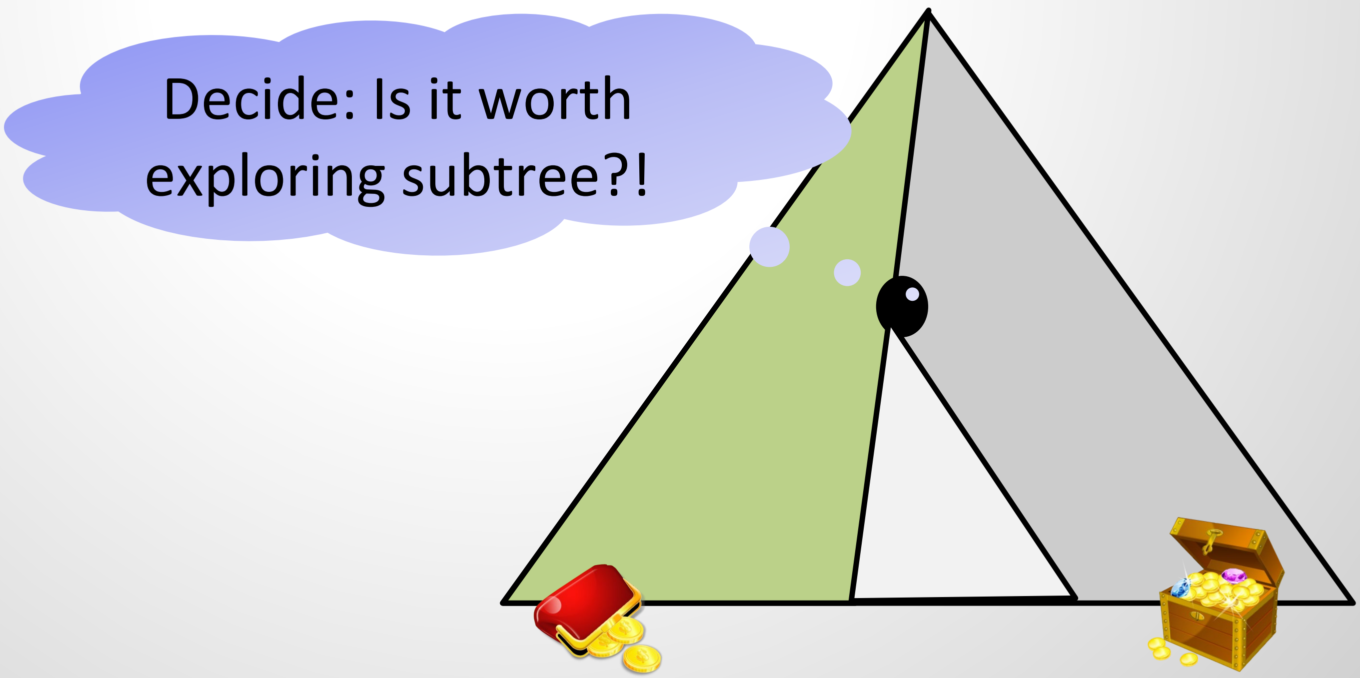


Can we at least formulate a “fast” MIP?

- Recall: Mixed Integer Program (MIP)
  - Linear objective function (e.g., minimize embedding footprint)
  - Linear constraints (e.g., do not violate capacity constraints)

- Solved, e.g., with branch-and-bound search tree

Usual procedure:

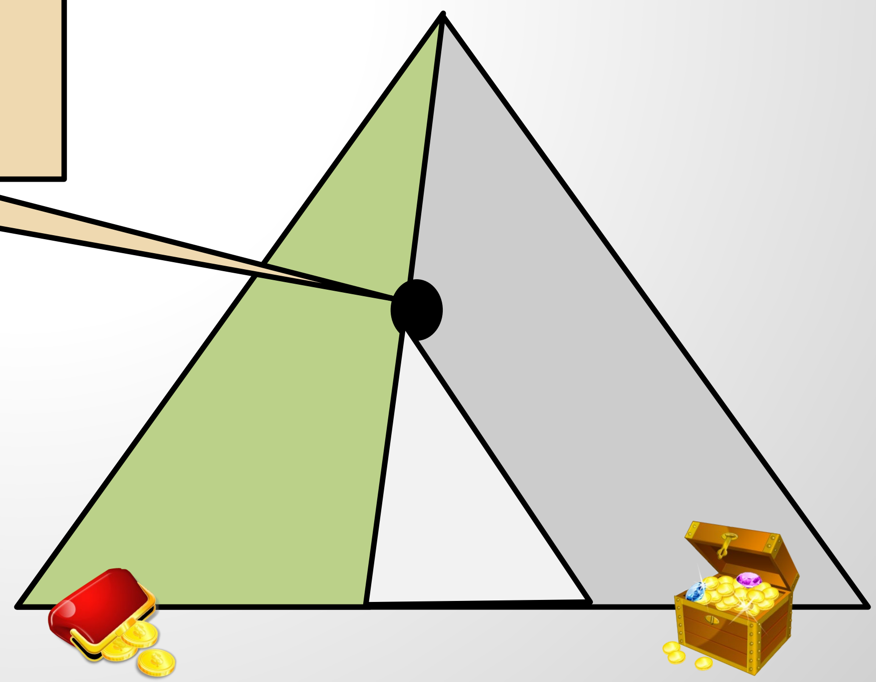


Can we at least formulate a “fast” MIP?

- Recall: Mixed Integer Program (MIP)
  - Linear objective function (e.g., minimize embedding footprint)
  - Linear constraints (e.g., do not violate capacity constraints)

- Solved, e.g., with branch-and-bound search tree

Usual trick: Relax! Solve LP (fast!), and if **relaxed solution** (more general!) **not better** then best solution so far: skip it!



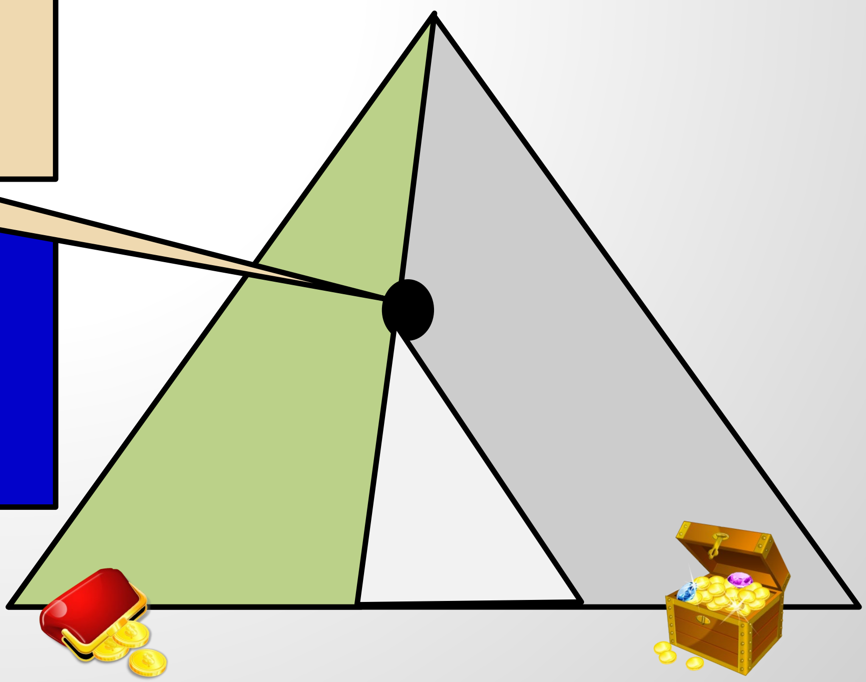
Can we at least formulate a "fast" MIP?

- Recall: Mixed Integer Program (MIP)
  - Linear objective function (e.g., minimize embedding footprint)
  - Linear constraints (e.g., do not violate capacity constraints)

- Solved, e.g., with branch-and-bound search tree

Usual trick: Relax! Solve LP (fast!), and if **relaxed solution** (more general!) **not better** then best solution so far: skip it!

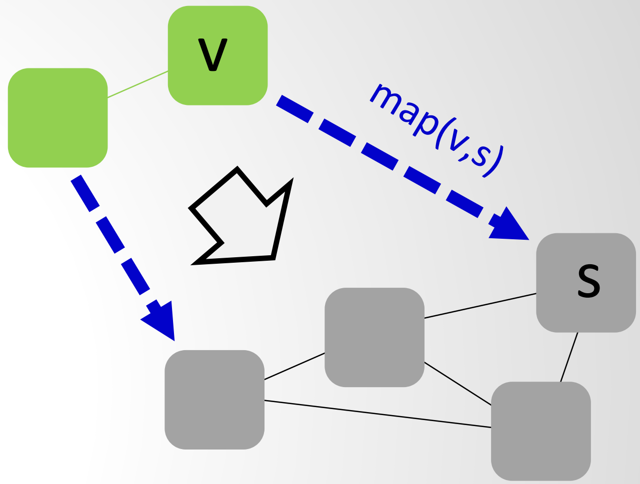
Bottomline: If MIP provides «good relaxations», large parts of the search space can be pruned.

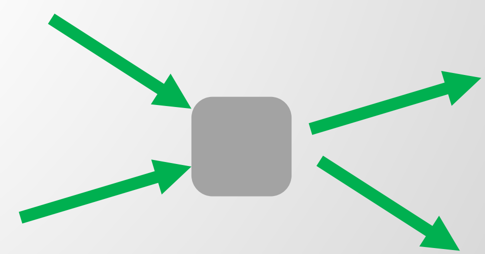


Can we at least formulate a "fast" MIP?

A typical MIP formulation:

- Introduce binary variables ***map(v,s)*** to map virtual nodes *v* to substrate node *s*
- Introduce flow variables for paths (say splittable for now)
- Ensure flow conservation: all flow entering a node must leave the node, unless it is the source or the destination





$$\Sigma_{u\text{->}v}\, f_{uv} = \Sigma_{v\text{->}w}\, f_{vw}$$

In Out

Can we at least formulate a “fast” MIP?

We get constraints like:

Assume bandwidth b requested from node s to node t.

In Out

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

What does this formula do and why is it correct?

Can we at least formulate a "fast" MIP?

We get constraints like:

Assume bandwidth b requested from node s to node t.

In Out

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

If map(s,v)=1, i.e., s mapped to v: so **flow starts at v**, and hence outgoing flow must be larger than incoming flow (plus b).

Can we at least formulate a “fast” MIP?

We get constraints like:

Assume bandwidth b requested from node s to node t.

In Out

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

If map(s,v)=0 and map(t,v)=0, i.e., **v is along the path** from s to t: then we have **flow conservation**: outgoing flow must equal incoming flow (here ≥, objective function will remove unnecessary flow).

Can we at least formulate a “fast” MIP?

We get constraints like:

Assume bandwidth b requested from node s to node t.

In Out

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

If map(t,v)=1, i.e., t mapped to v: so flow **terminates at node v**: so **no constraint**: minus infinity (but objective function will remove unnecessary flow).

Can we at least formulate a "fast" MIP?

We get constraints like:

Assume bandwidth b requested from node s to node t.

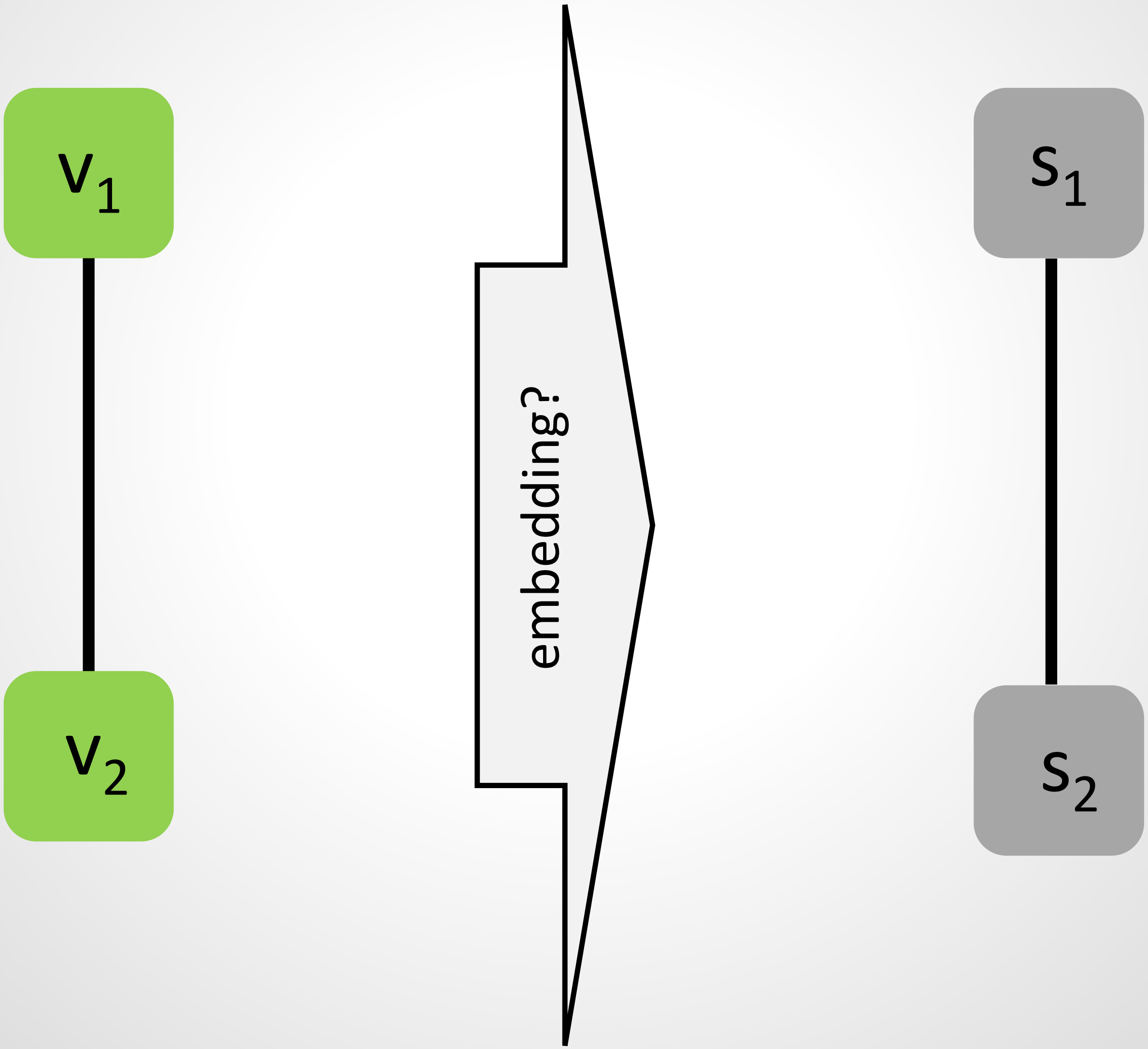
In Out

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

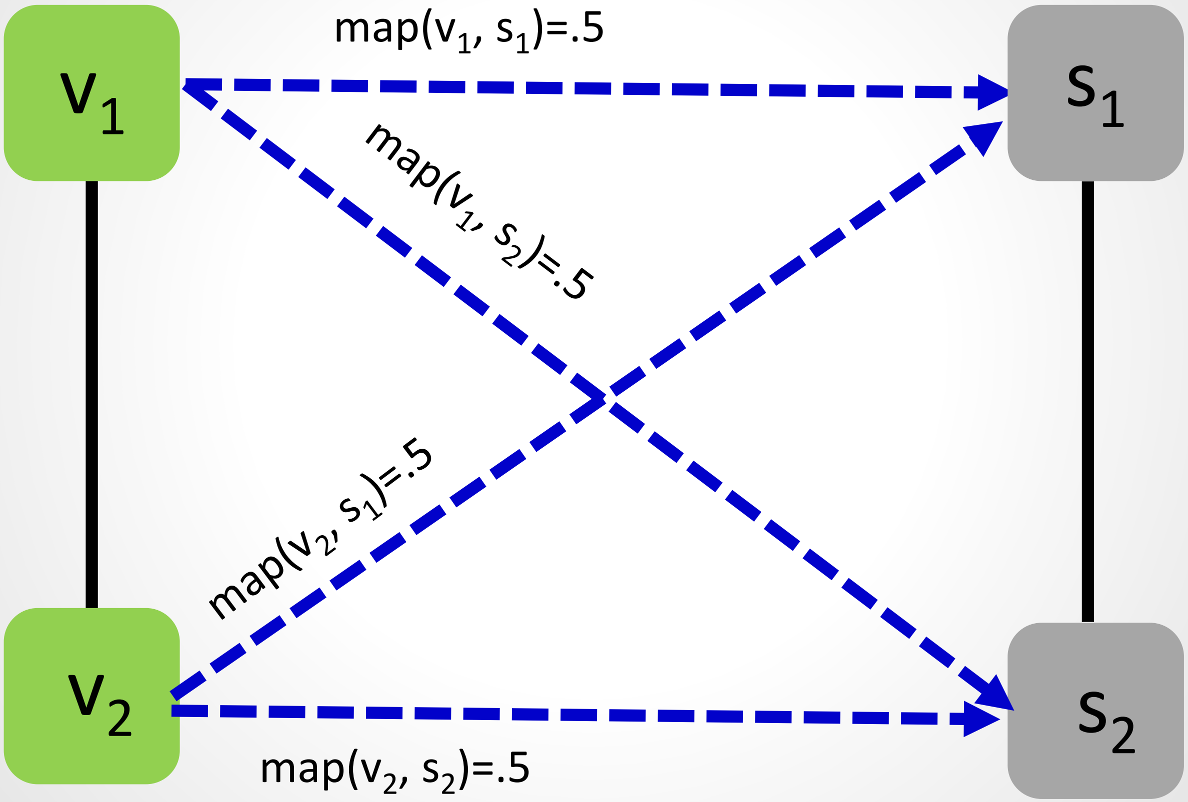
If map(t,v)=1, i.e., t mapped to v: so flow **terminates at node v**: so **no constraint**: minus infinity (but objective function will remove unnecessary flow).

Will such a MIP provide effective pruning?

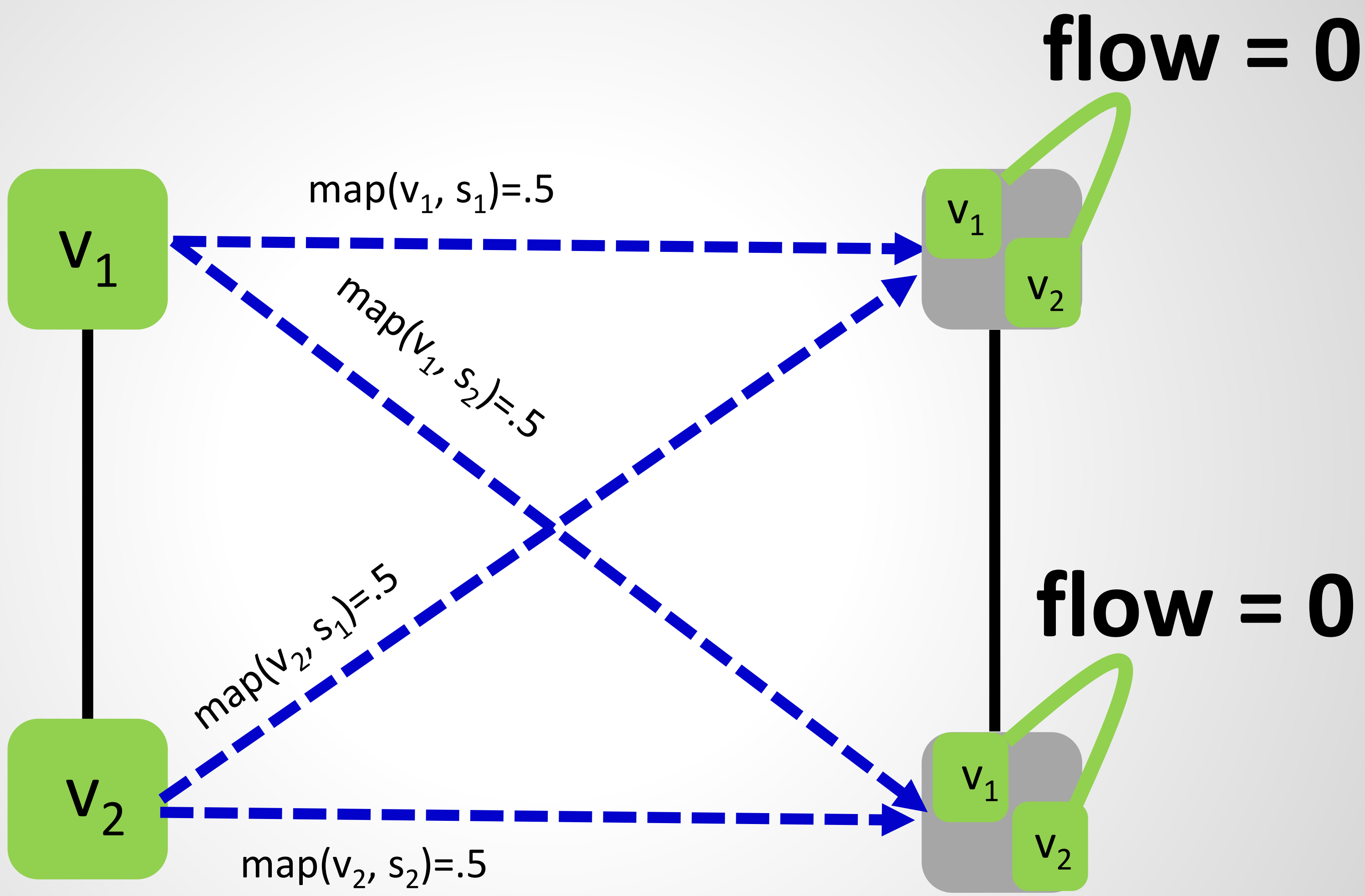
What will happen in this example?



What will happen in this example?



What will happen in this example?



Minimal flow = 0: fulfills flow conservation! Relaxation useless: does not provide any lower bound or indication of good mapping!

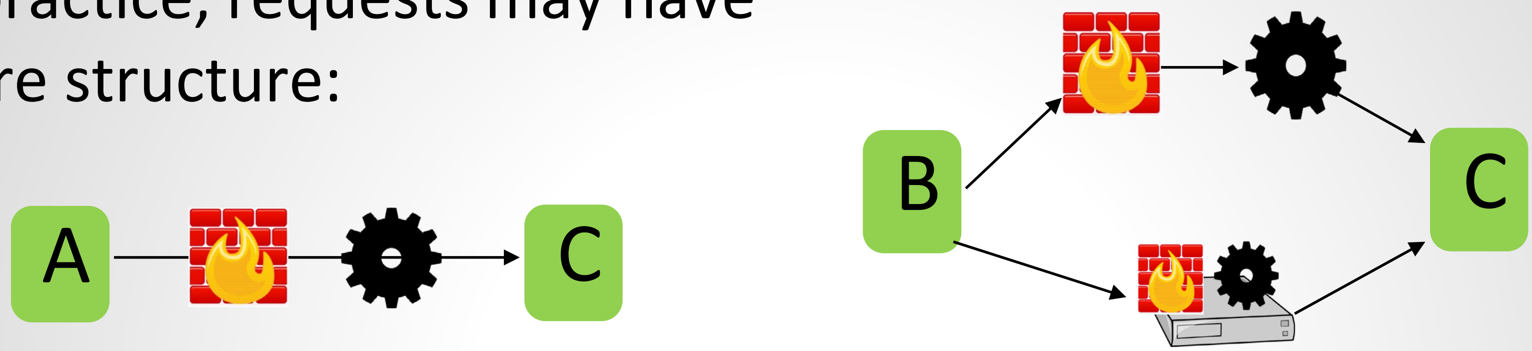
Wait a minute!

These problems need to be solved!

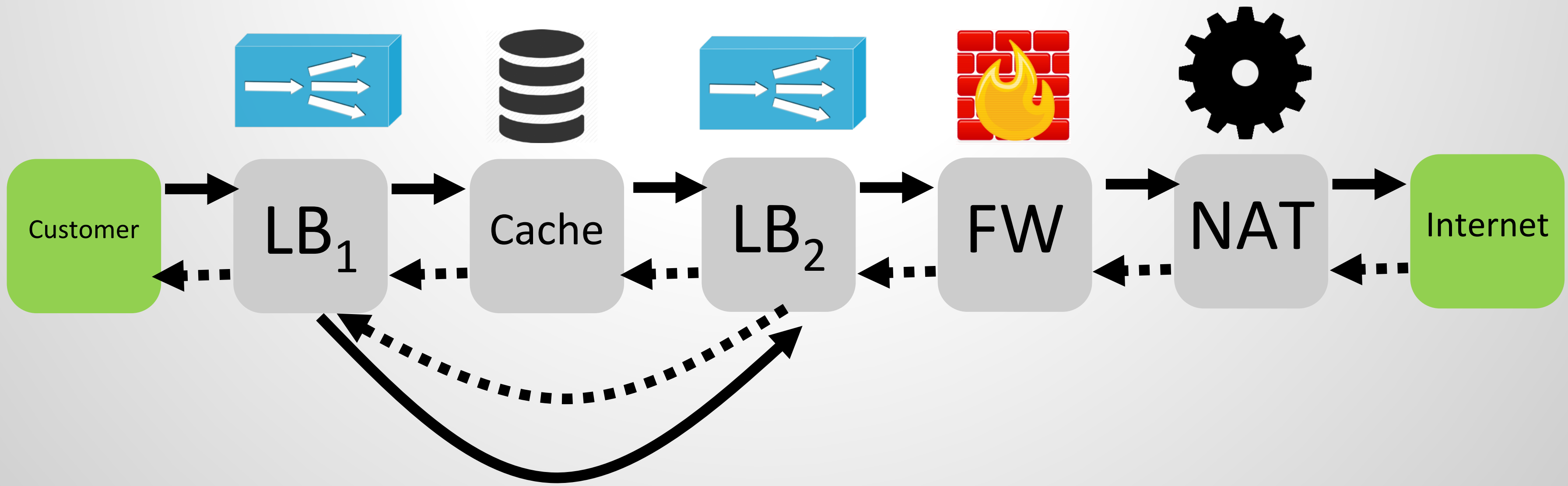
And they often can, even with guarantees.

Theory vs Practice:
In Practice There is Hope!

In practice, requests may have more structure:

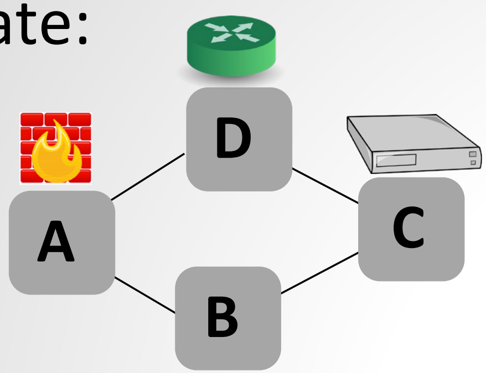


Even this beast:

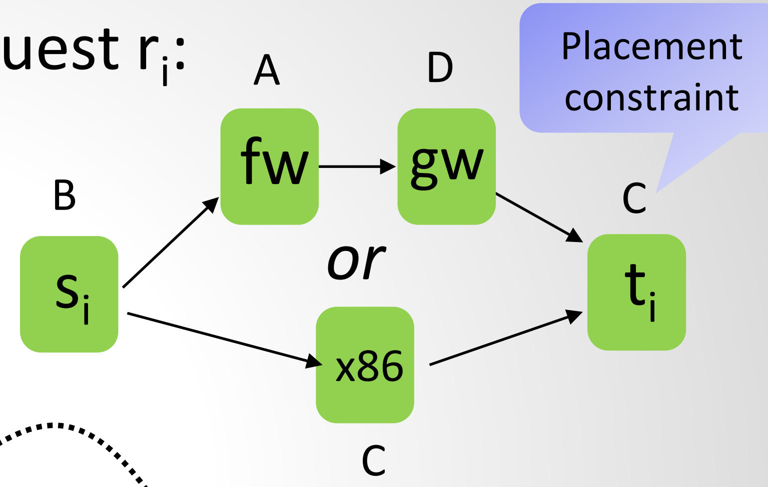


But In Theory There is Hope Too: Approximations! Product graphs and randomized rounding

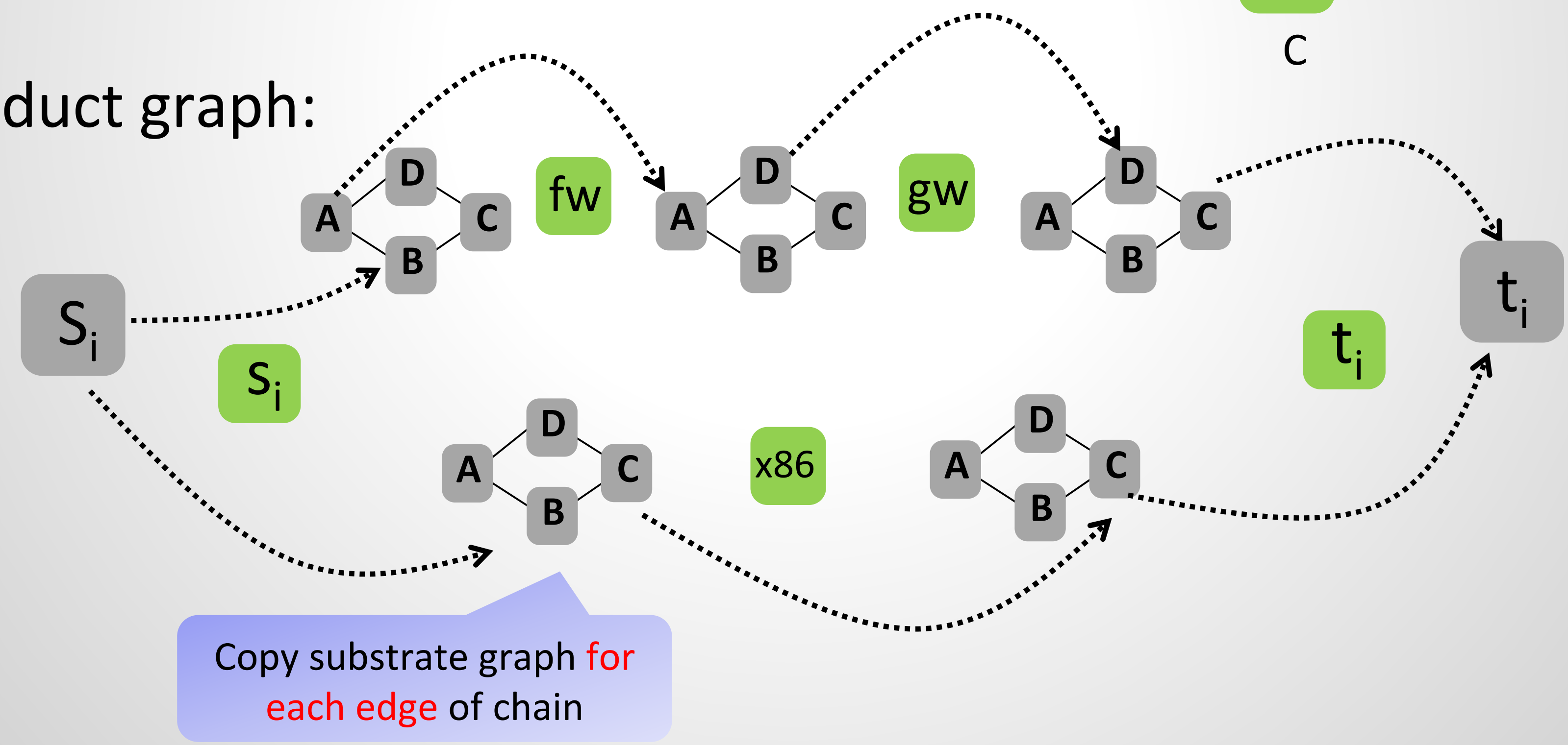
Substrate:



i^{th} request r_i:

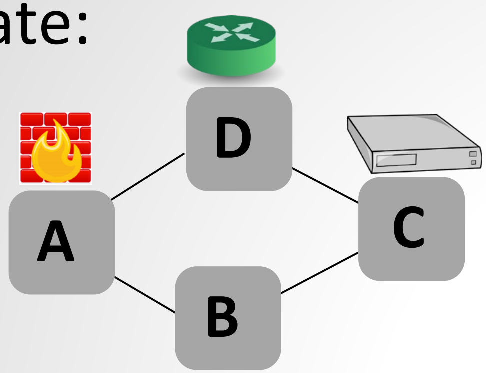


Product graph:

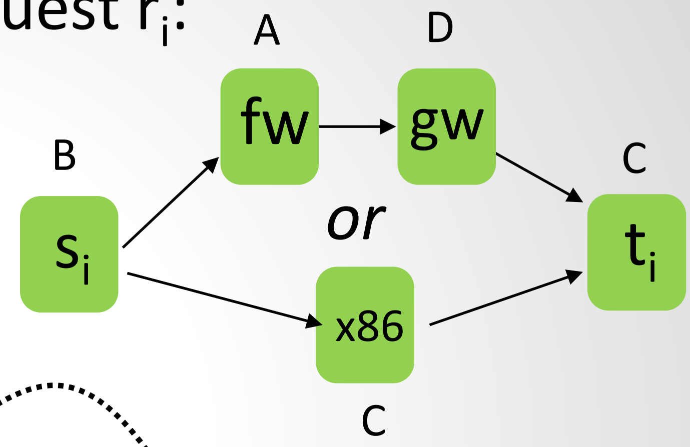


But In Theory There is Hope Too: Approximations!
Product graphs and randomized rounding

Substrate:

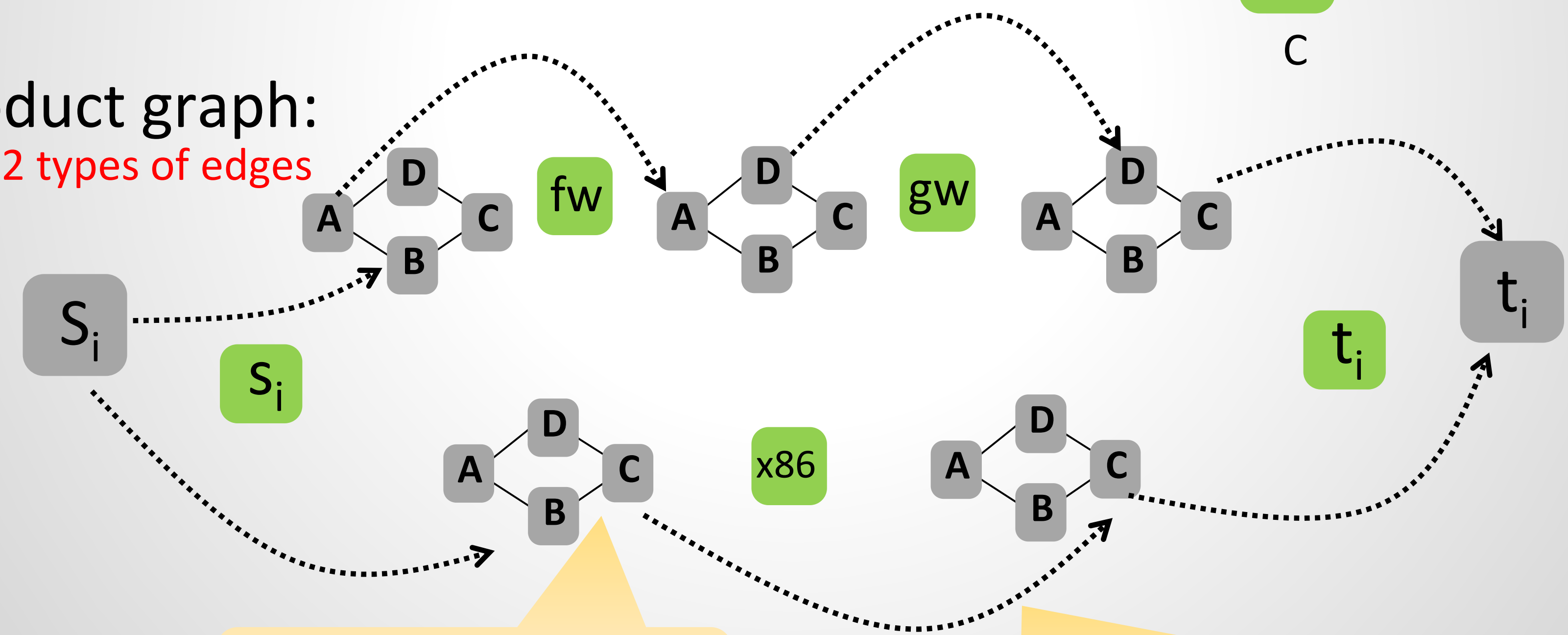


i^{th} request r_i:



Product graph:

with 2 types of edges

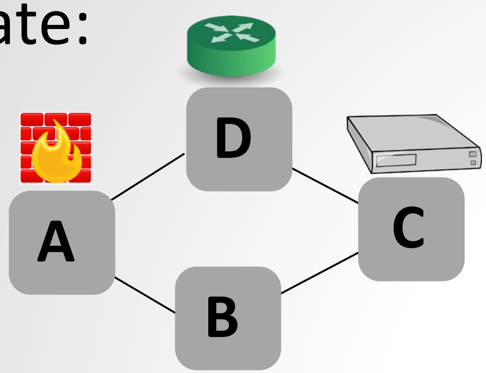


Routing edge: graph edge on same layer

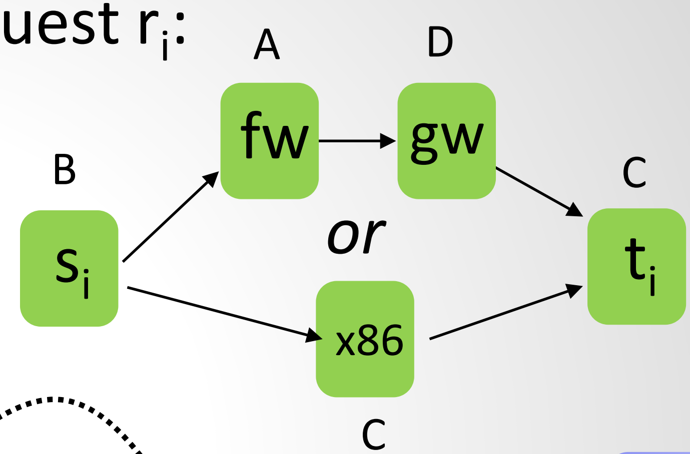
Processing edge: processing happens on C: connect C to C in next layer!

But In Theory There is Hope Too: Approximations!
Product graphs and randomized rounding

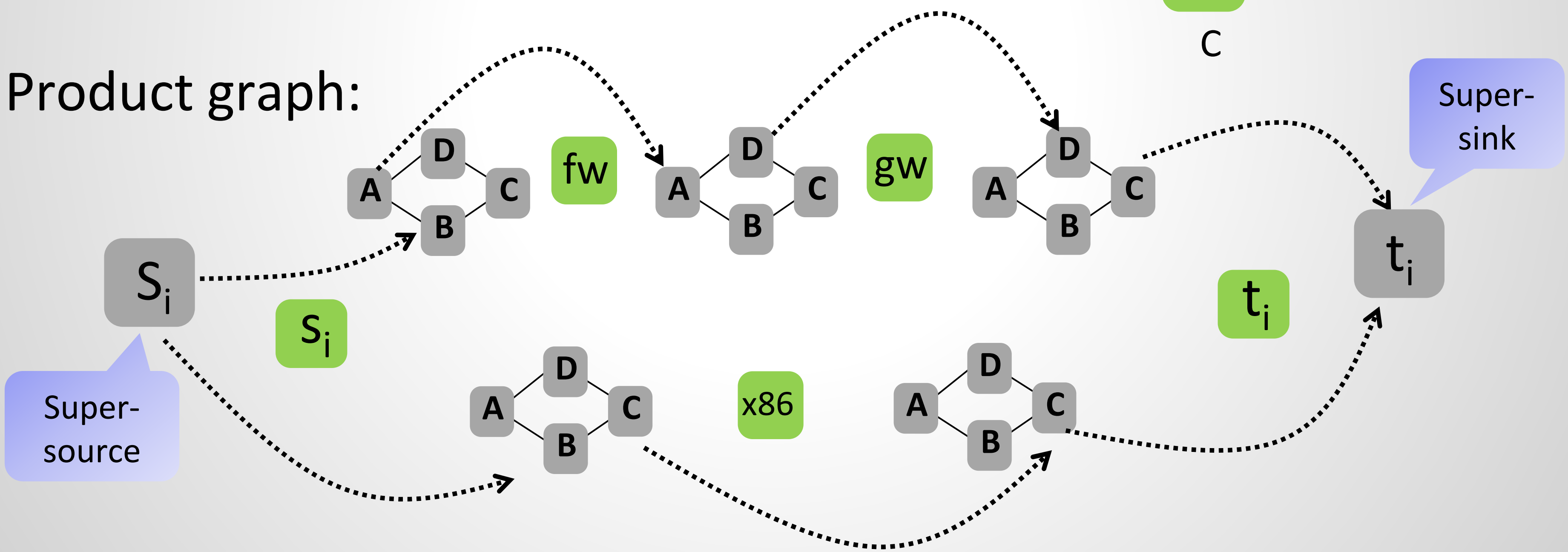
Substrate:



i^{th} request r_i:

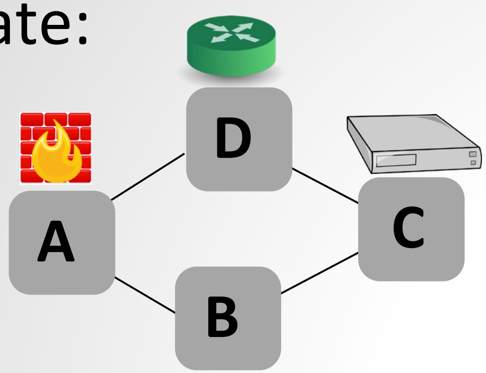


Product graph:

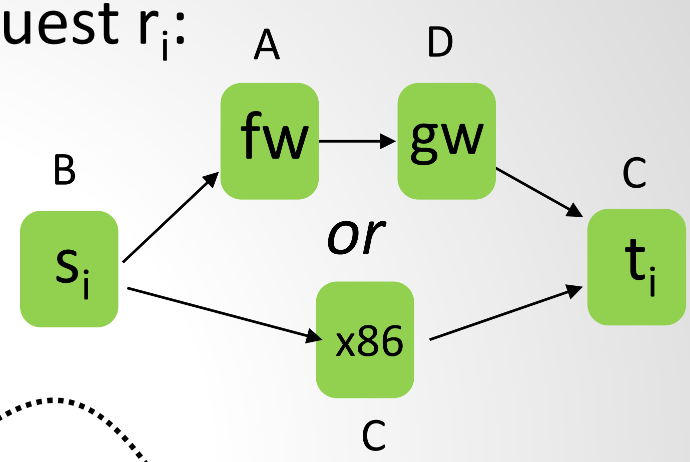


But In Theory There is Hope Too: Approximations!
Product graphs and randomized rounding

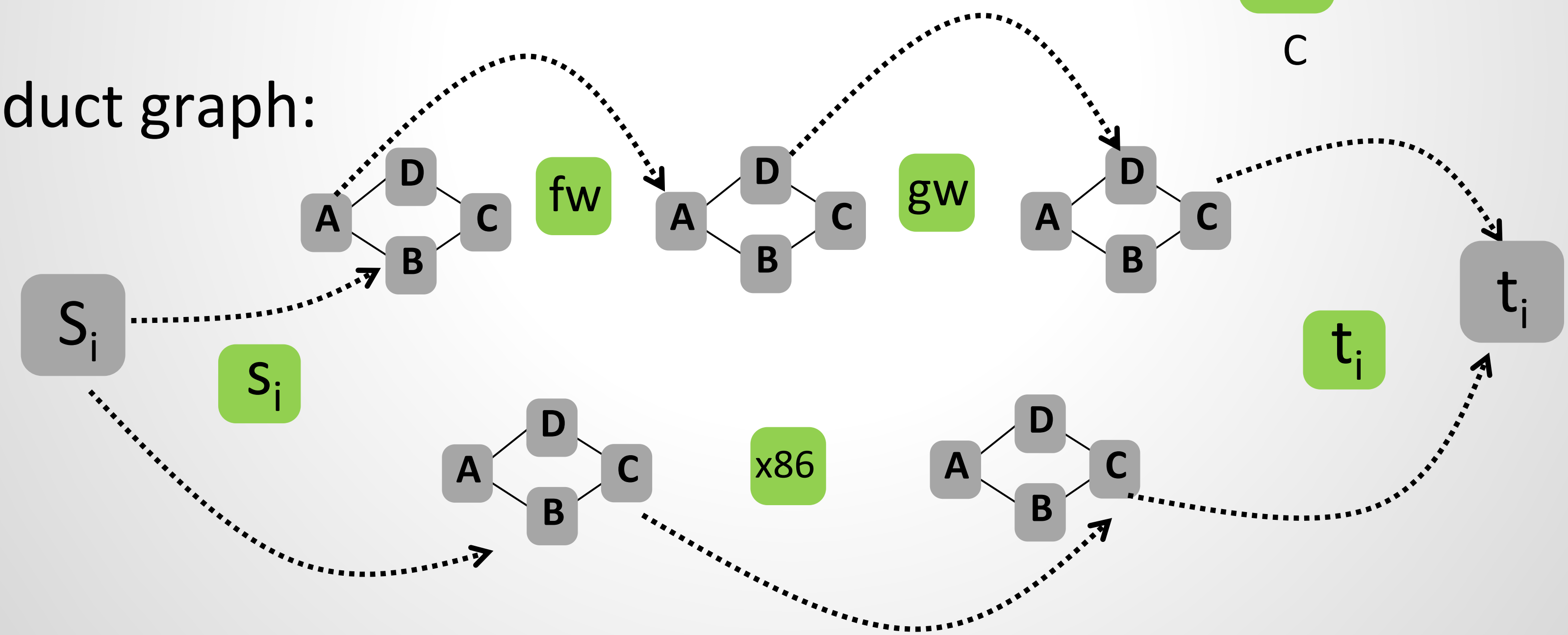
Substrate:



i^{th} request r_i:



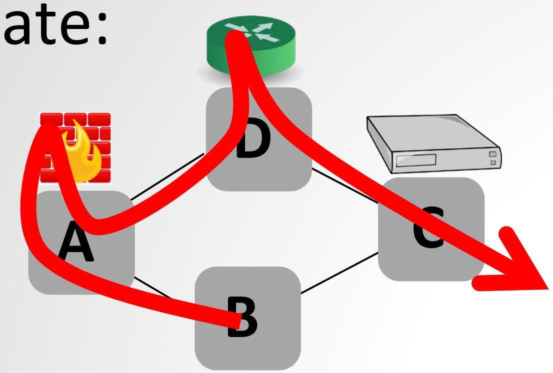
Product graph:



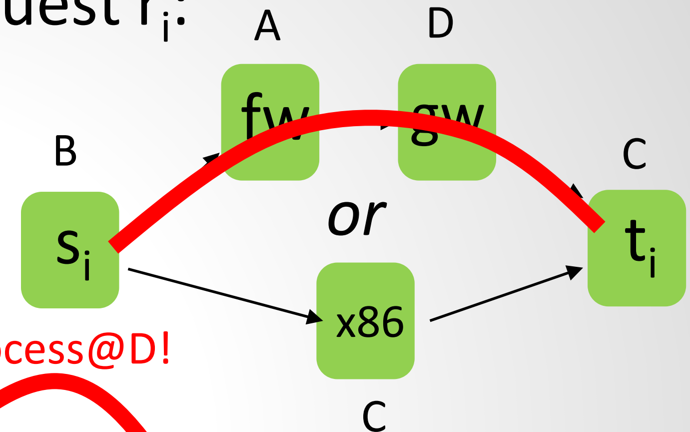
Any (s_i, t_i) flow presents a route of the request r_i!

But In Theory There is Hope Too: Approximations!
Product graphs and randomized rounding

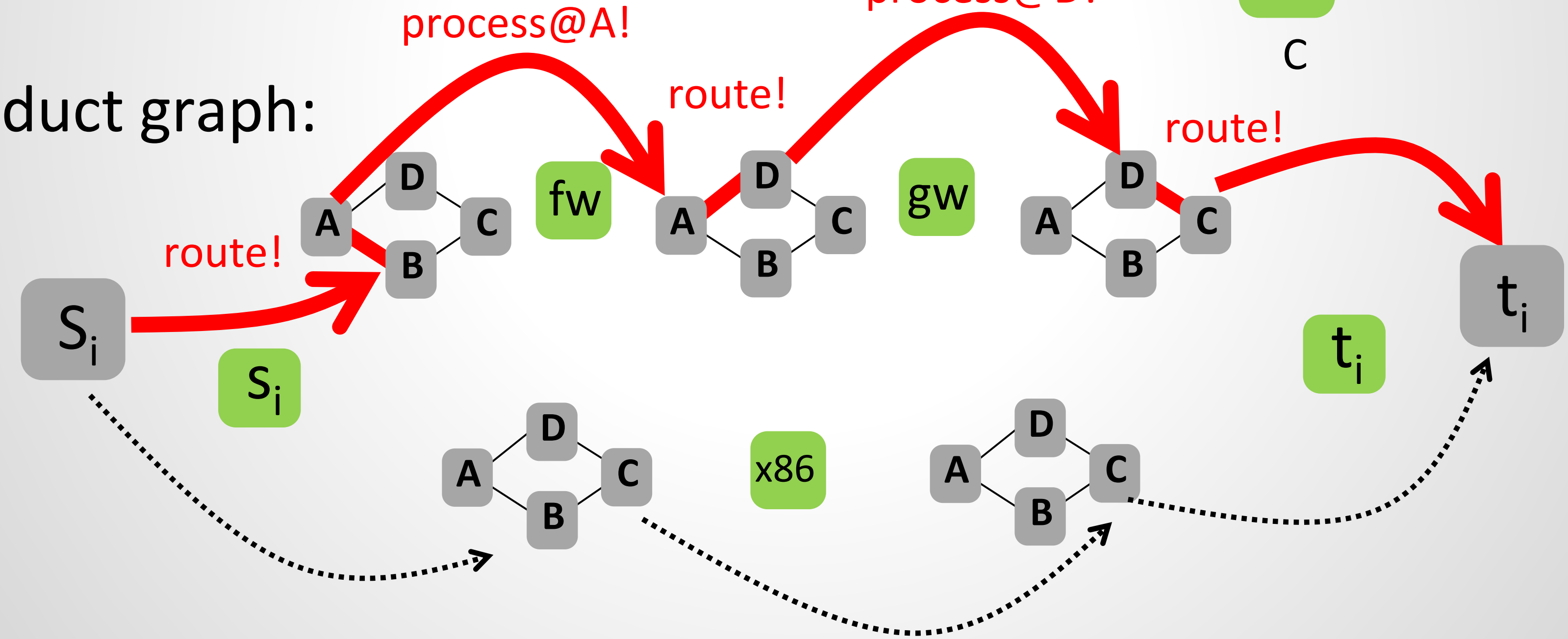
Substrate:



i^{th} request r_i:



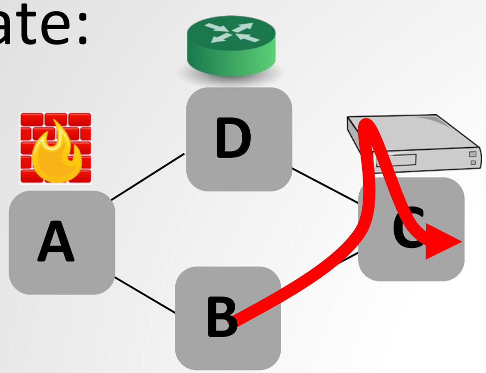
Product graph:



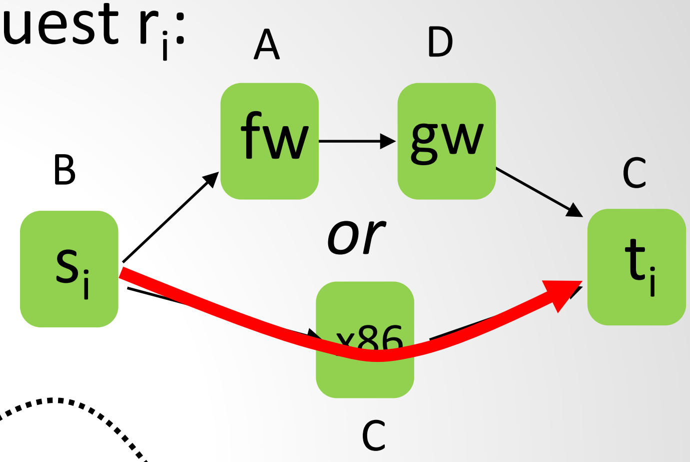
Any (s_i,t_i) flow presents a route of the request r_i!

But In Theory There is Hope Too: Approximations! Product graphs and randomized rounding

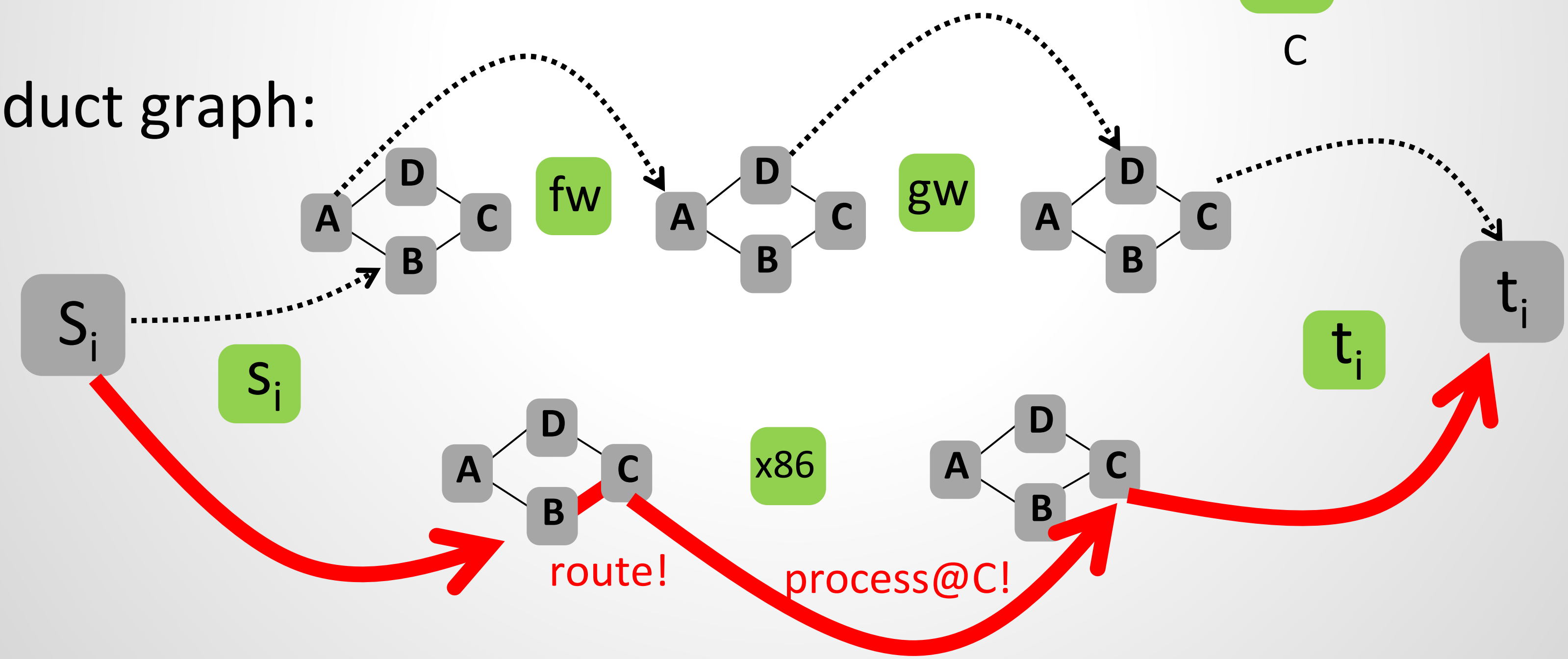
Substrate:



i^{th} request r_i:



Product graph:



Any (s_i,t_i) flow presents a route of the request r_i!

But In Theory There is Hope Too: Approximations!

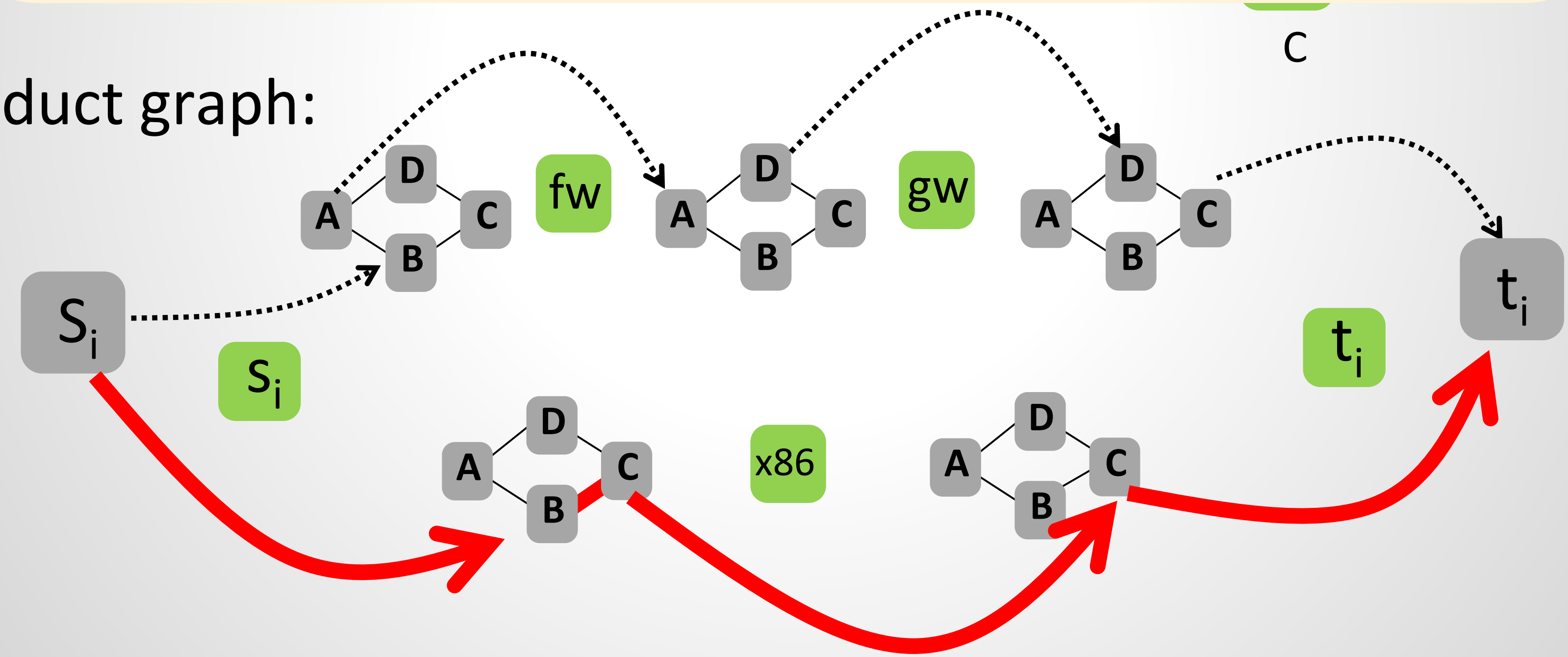
Product graphs and randomized rounding

Su

This problem can be solved using mincost unsplittable multi-commodity flow (approximation) algorithms (e.g., randomized rounding).

C

Product graph:



Any (s_i,t_i) flow presents a route of the request r_i!

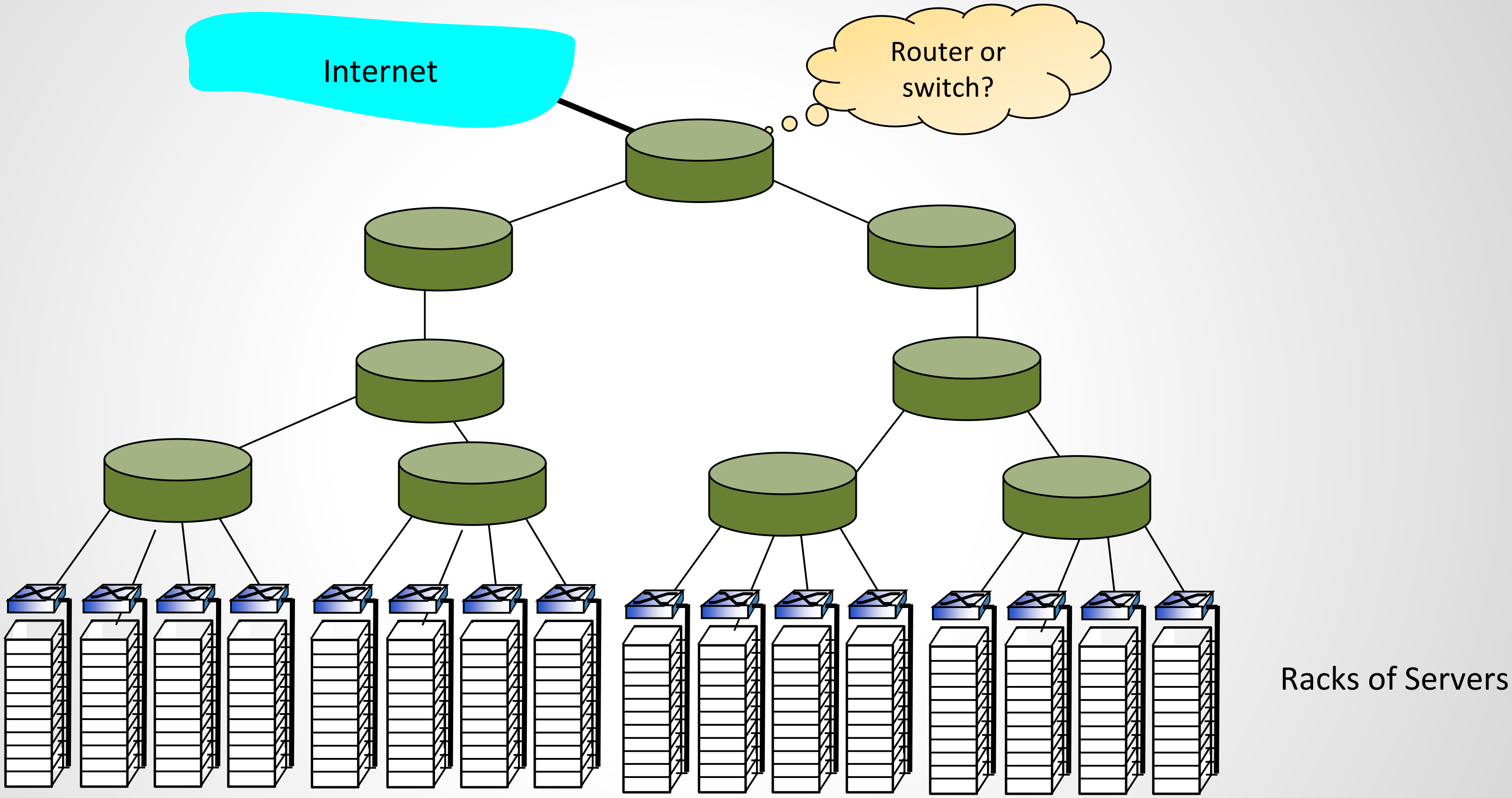
Invitation: A Roadtrip Through The Opportunities and Challenges of Network Isolation

- ❑ Opportunities
  - ❑ Algorithmic opportunities
  - ❑ **Technological opportunities**
- ❑ Challenges
  - ❑ Modelling challenges
  - ❑ Security challenges
- ❑ A perspective how AI can improve slicing efficiency and security

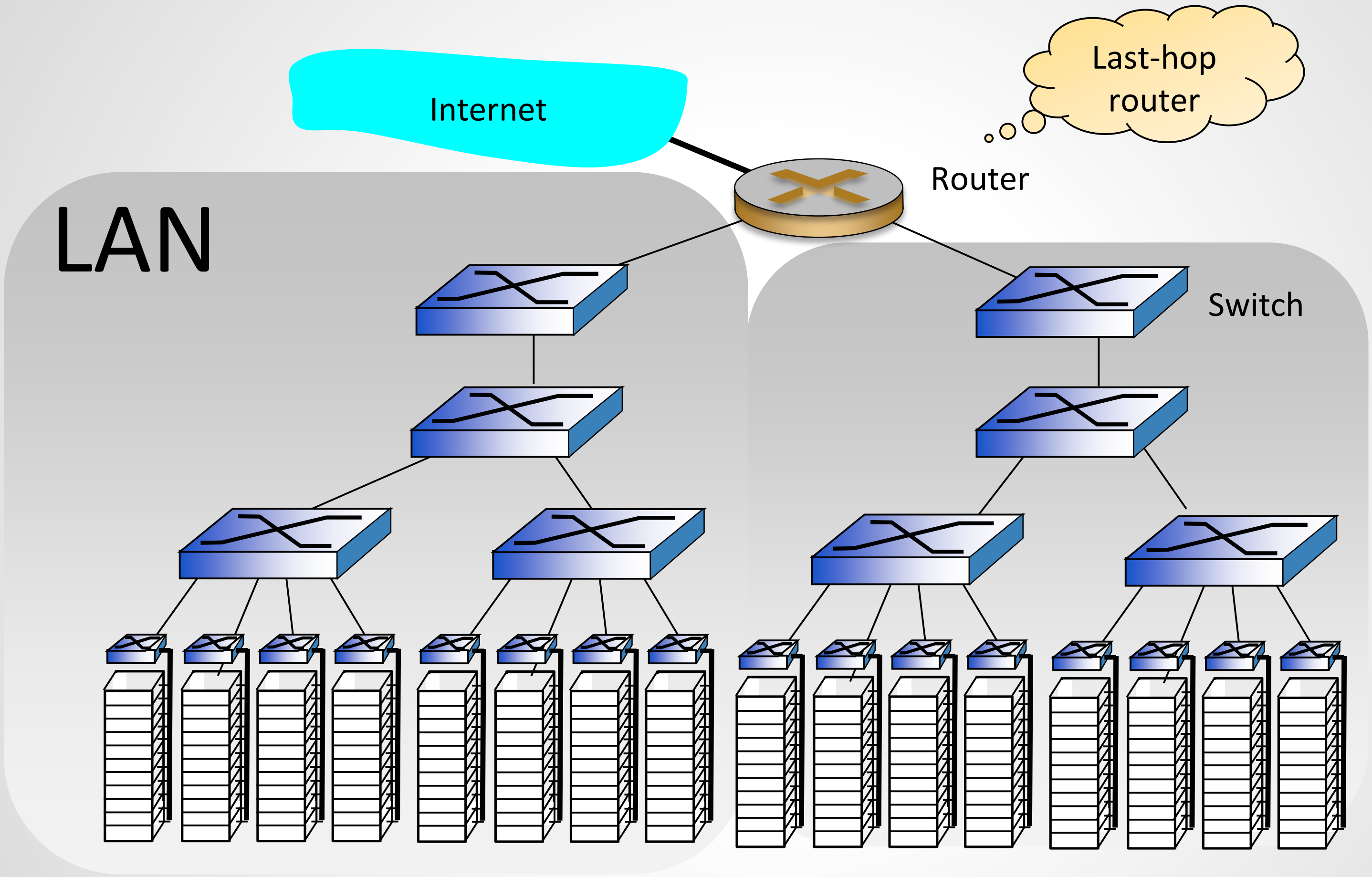
Road map 1927: Arizona and New Mexico

Example: Isolation in Datacenter

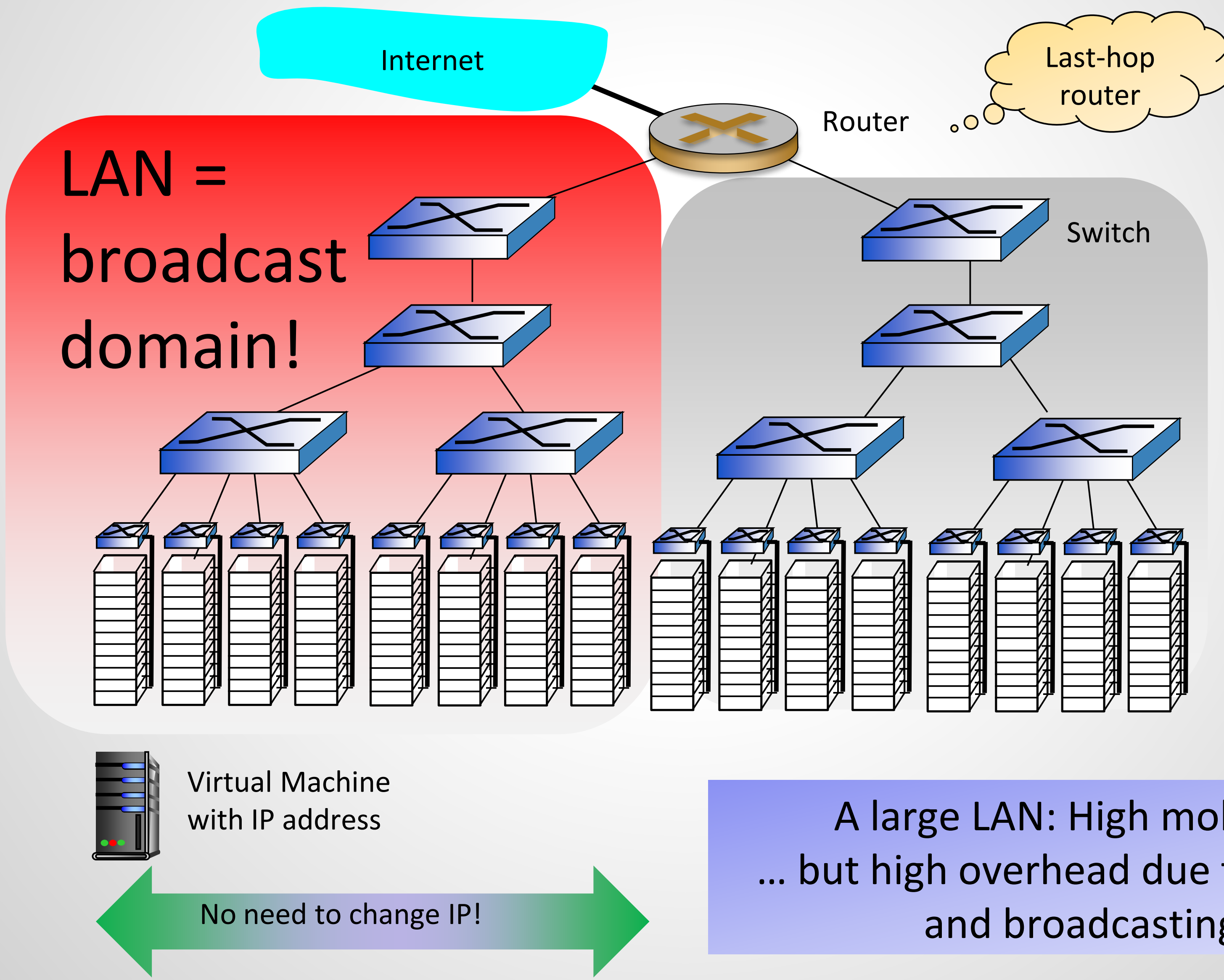
Tradeoff, traditionally:



Architecture #1

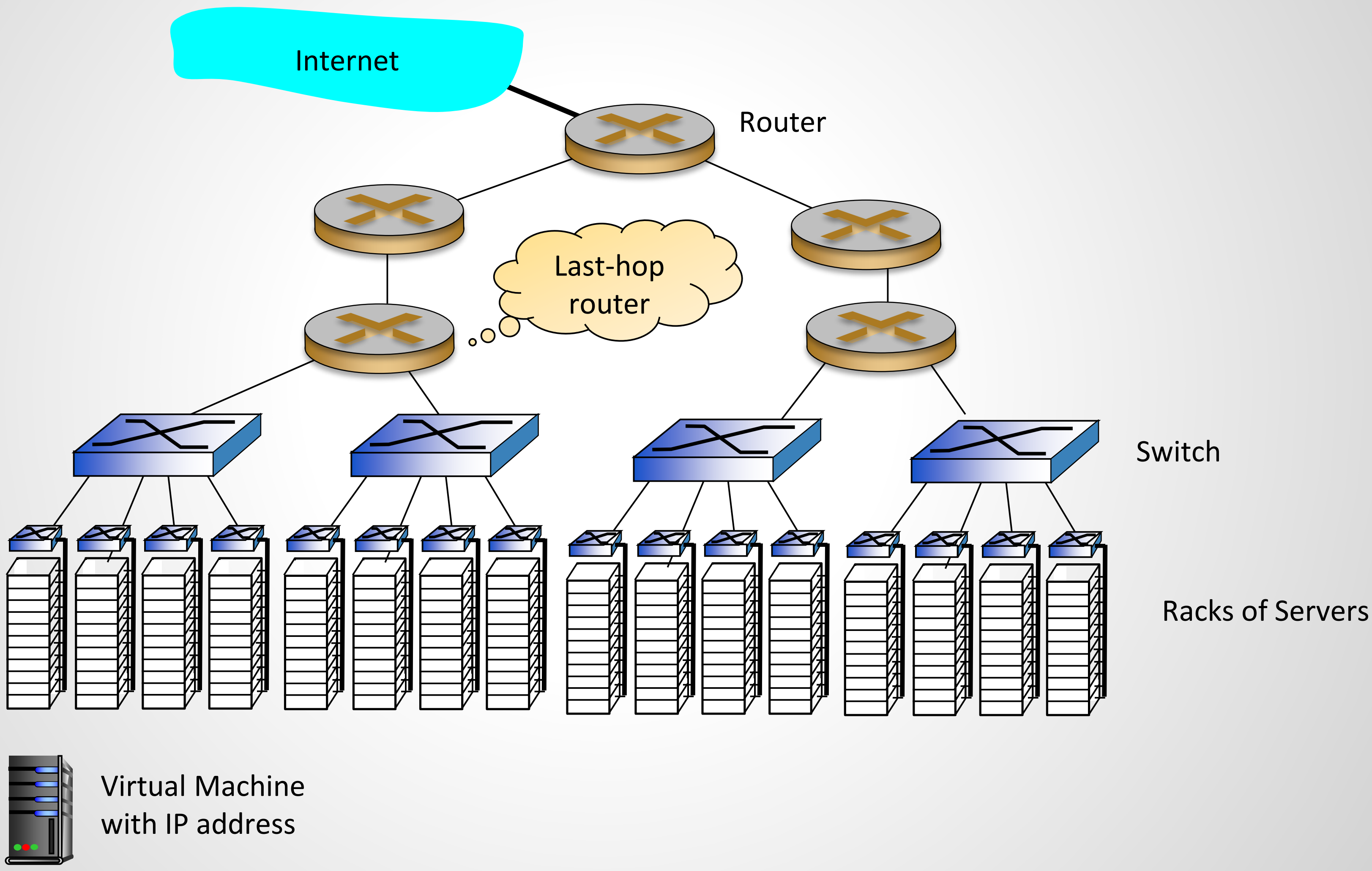


Architecture #1

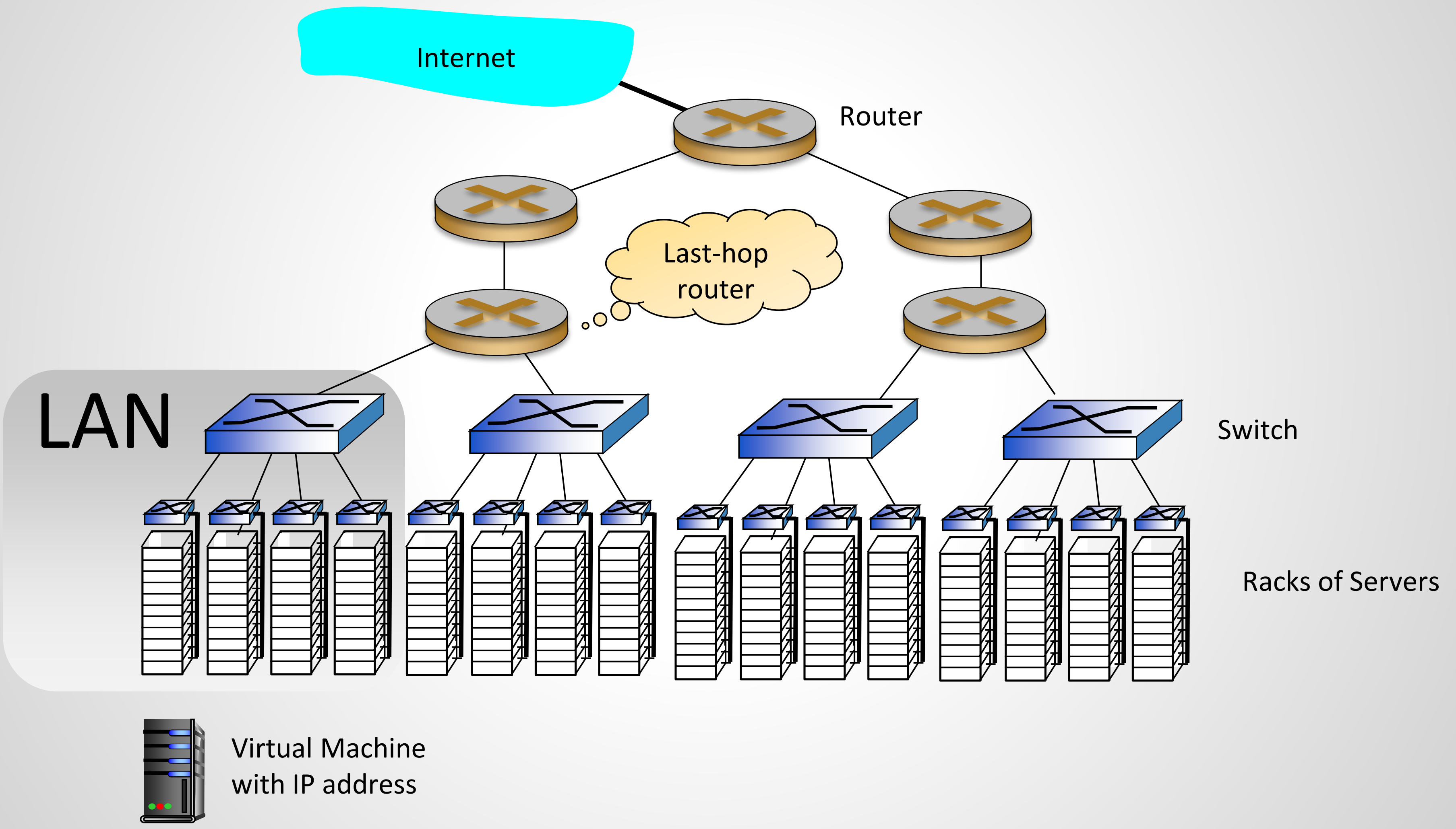


A large LAN: High mobility…
… but high overhead due to learning and broadcasting.

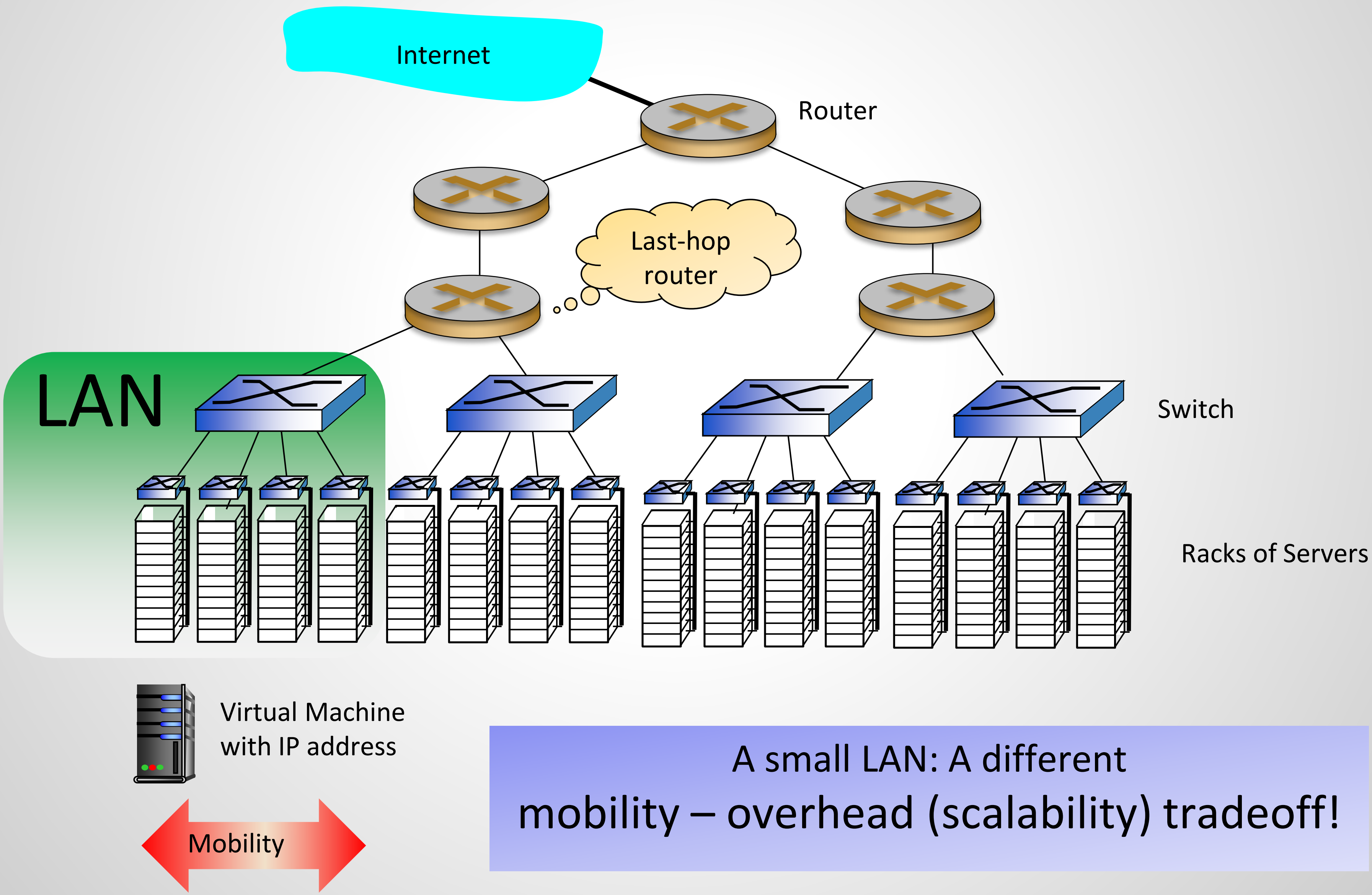
Architecture #2



Architecture #2

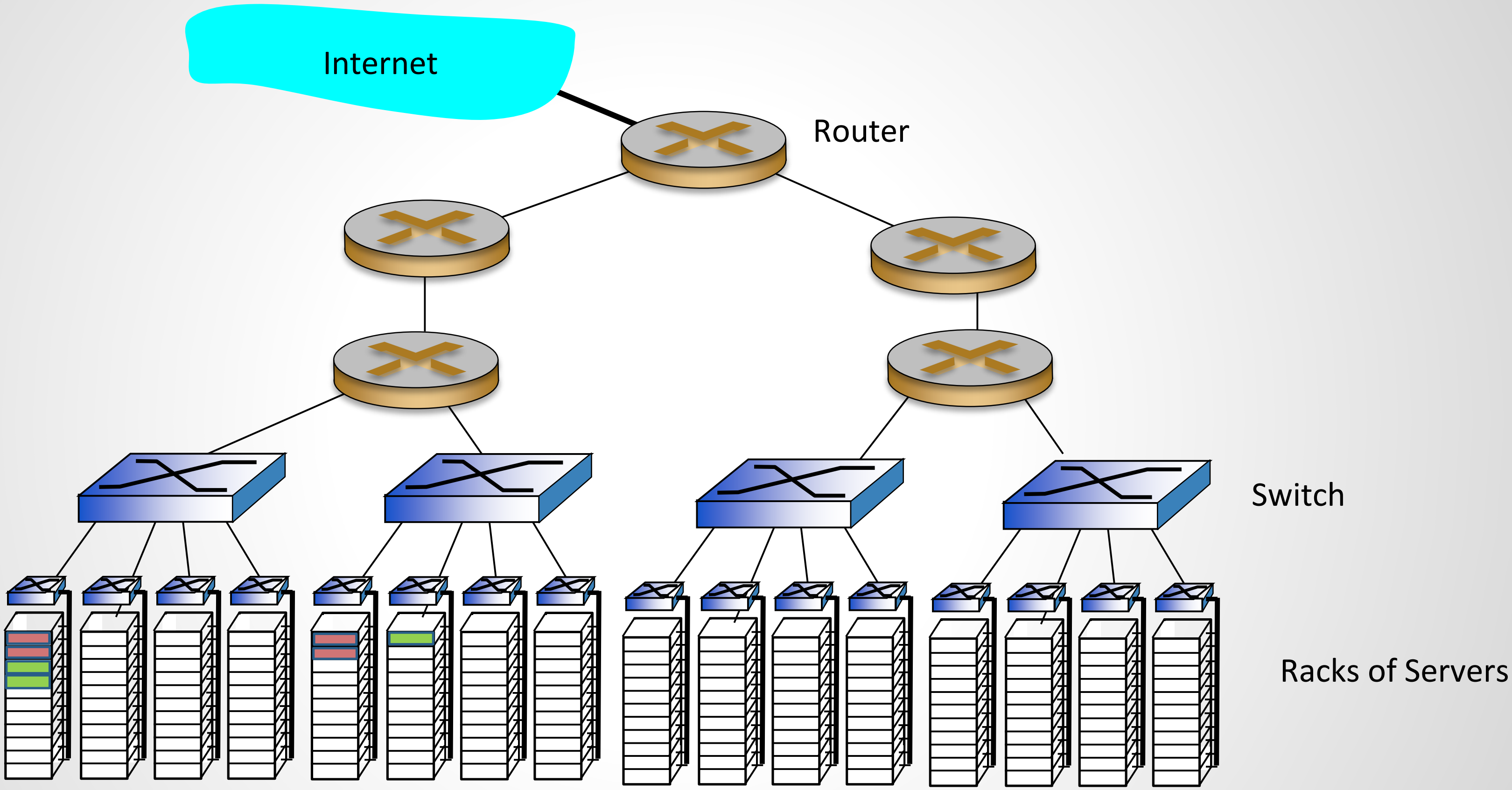


Architecture #2

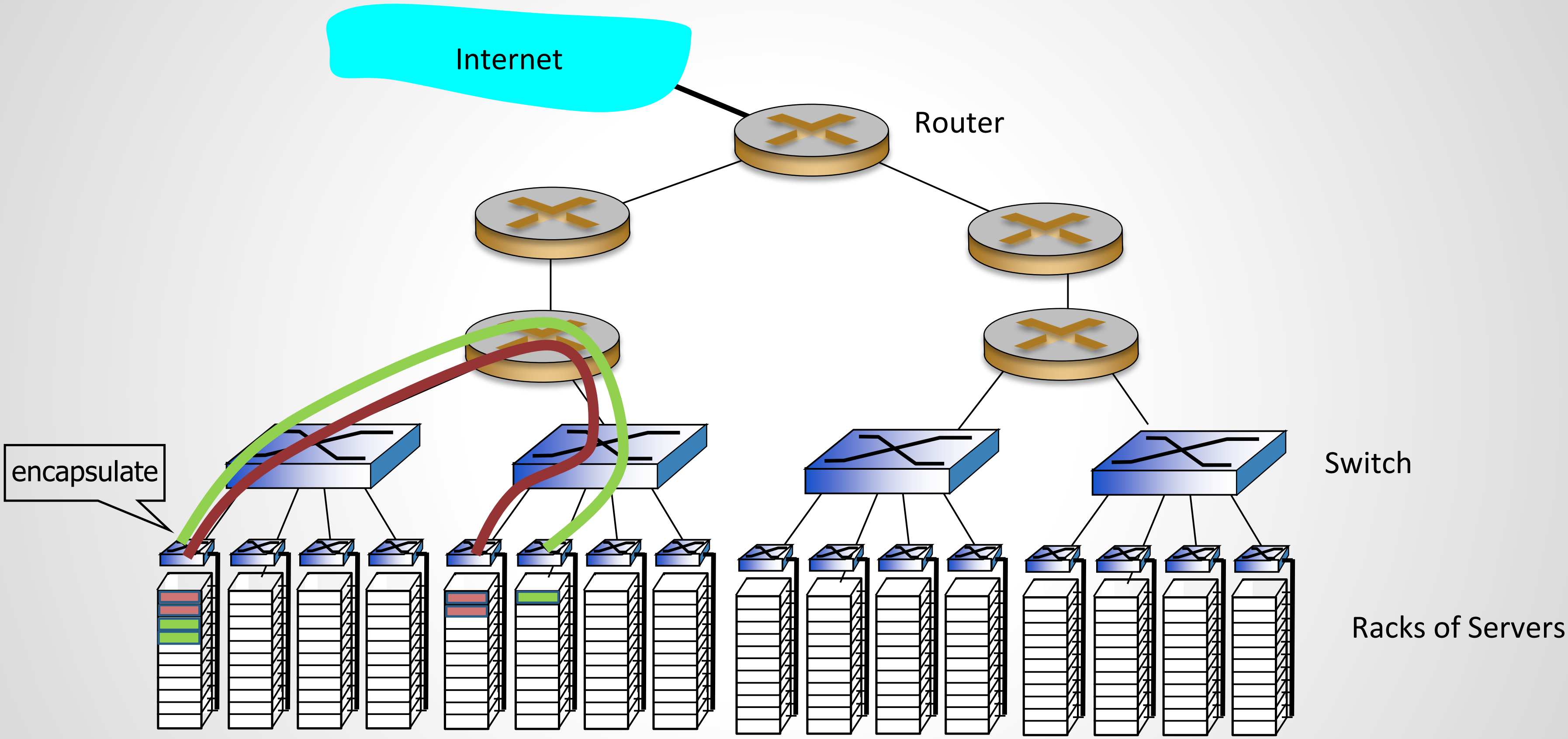


A small LAN: A different
mobility – overhead (scalability) tradeoff!

Virtualization Technologies: Isolation of Tenants



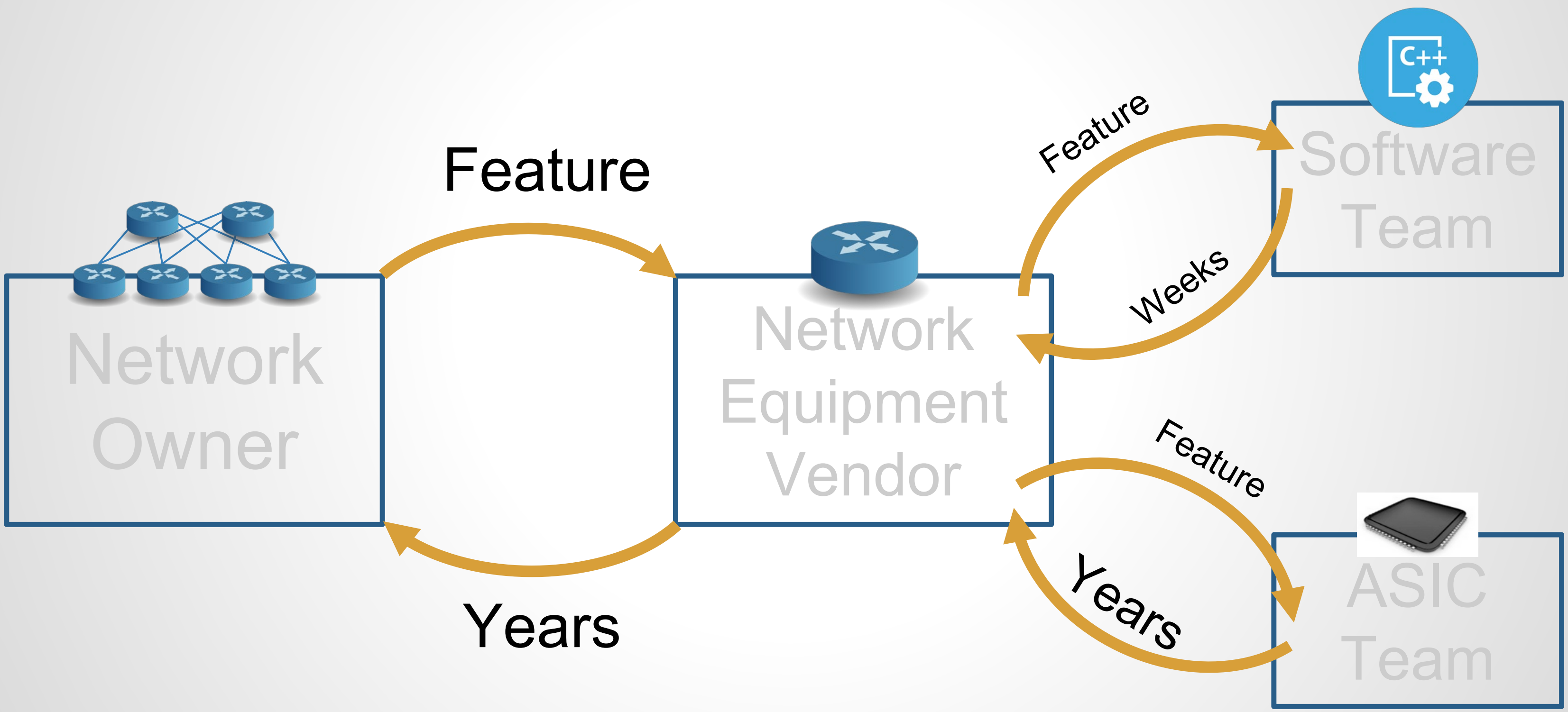
Virtualization Technologies: Isolation of Tenants



Network virtualization: VLANs, VxLANs, tunneling, … *or SDN*!

In the Past, Introducing Virtualization Technologies Took Years

Example: VxLAN



In the Past: Slow Innovation

Operator says:

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

Vendor's answer:

Buy one of these!

In the Past: Slow Innovation

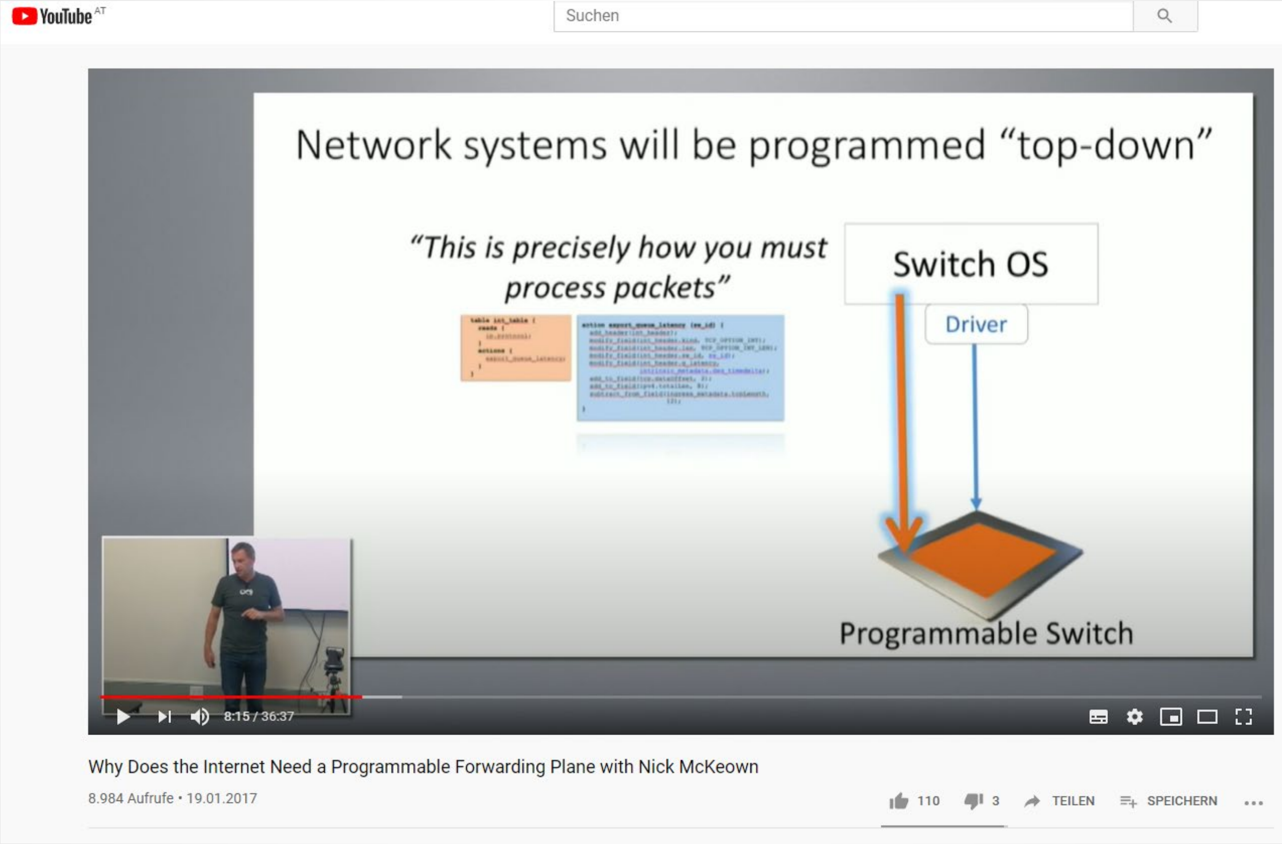
Operator says:

Vendor's answer:

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

We don't have that!

Opportunity: Softwarization, e.g., Programmable Dataplanes
Innovation at the Speed of Software Development



https://www.youtube.com/watch?v=zR88NIg3n3g

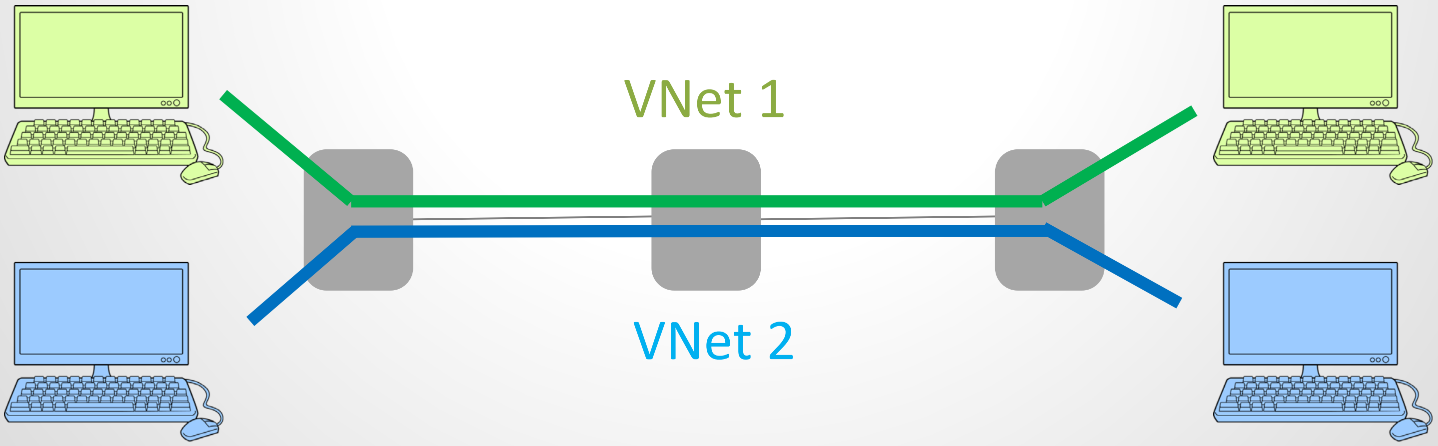
Invitation: A Roadtrip Through The Opportunities and Challenges of Network Isolation

- ❑ Opportunities
  - ❑ Algorithmic opportunities
  - ❑ Technological opportunities
- ❑ Challenges
  - ❑ **Modelling challenges**
  - ❑ Security challenges
- ❑ A perspective how AI can improve slicing efficiency and security

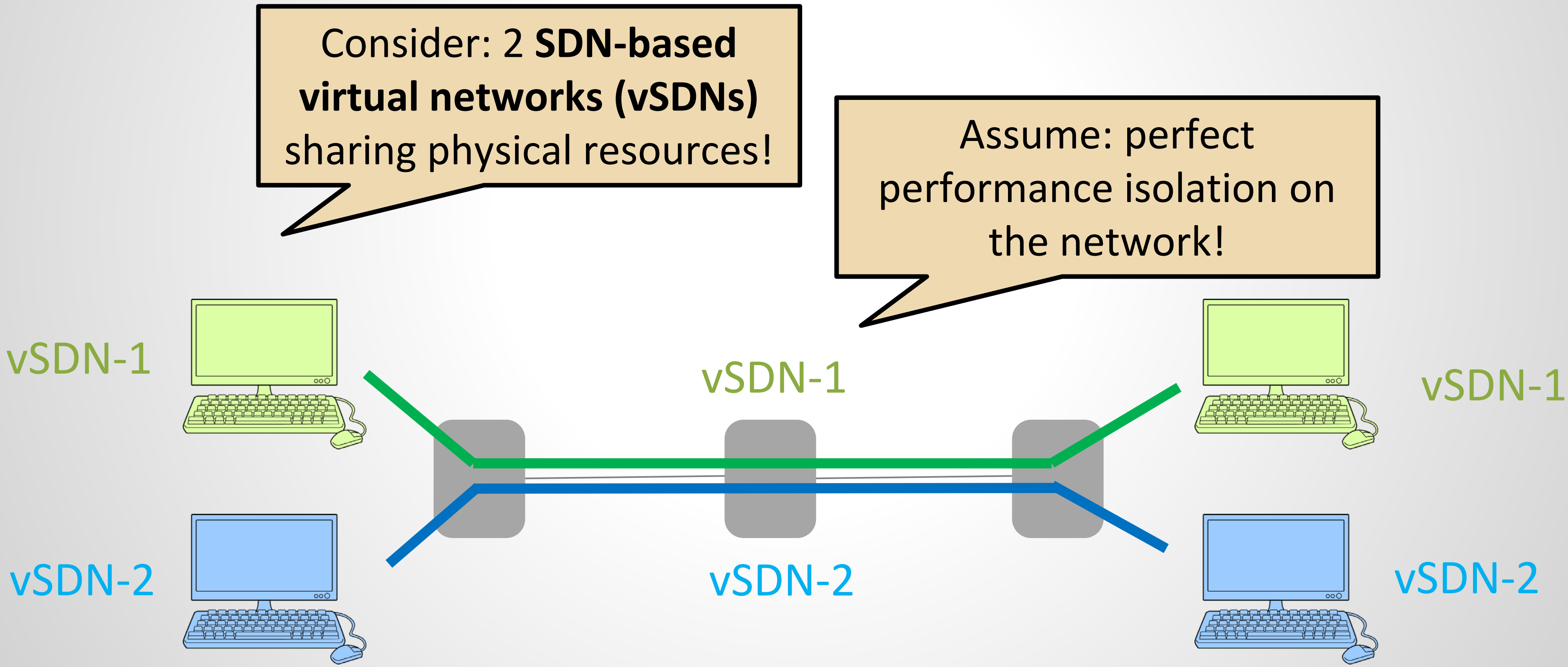
Road map 1927: Arizona and New Mexico

Models: Mind the Gap!

In theory land: bandwidth reservation for virtual networks = predictable performance

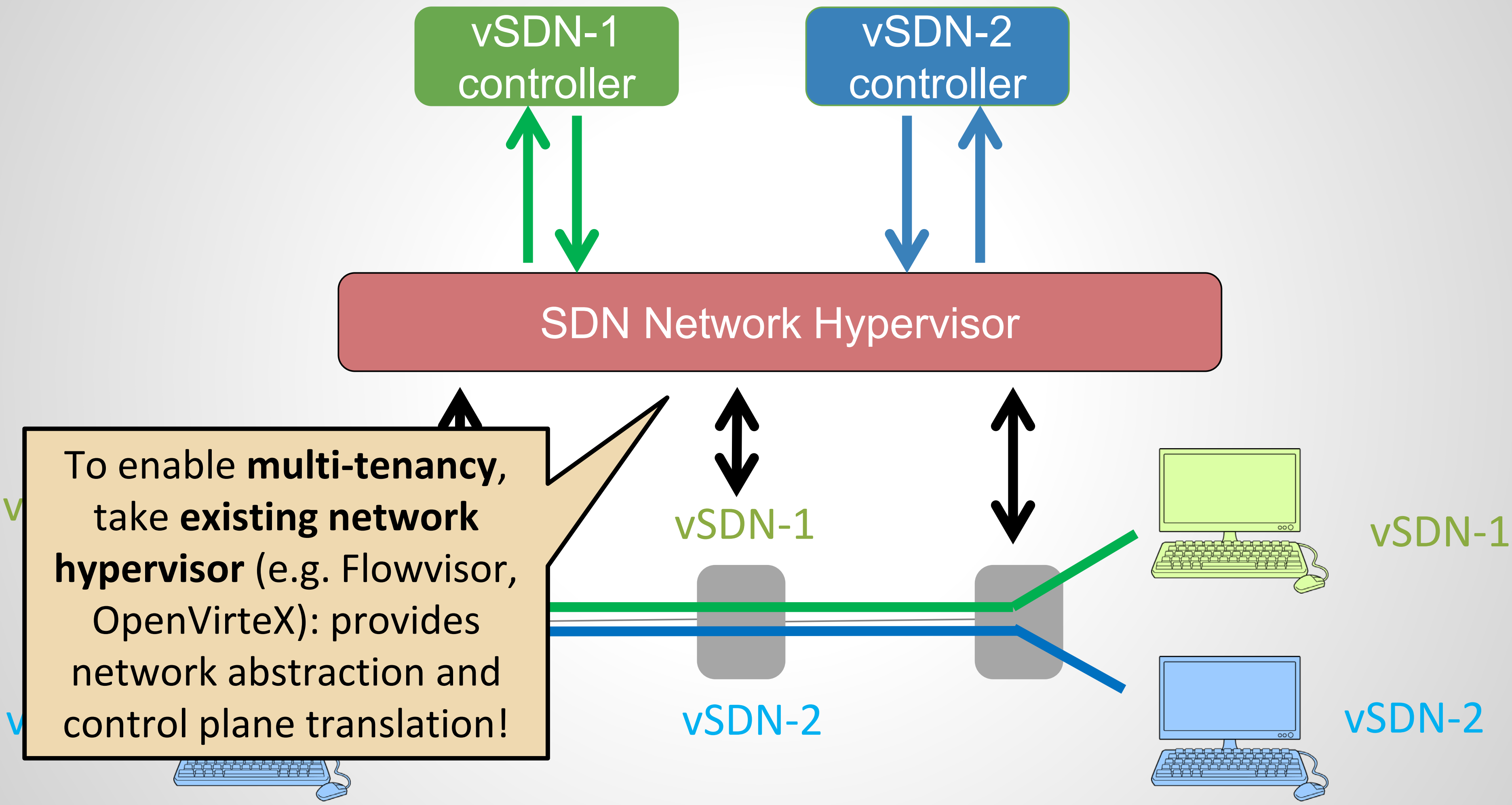


Models: Mind the Gap!



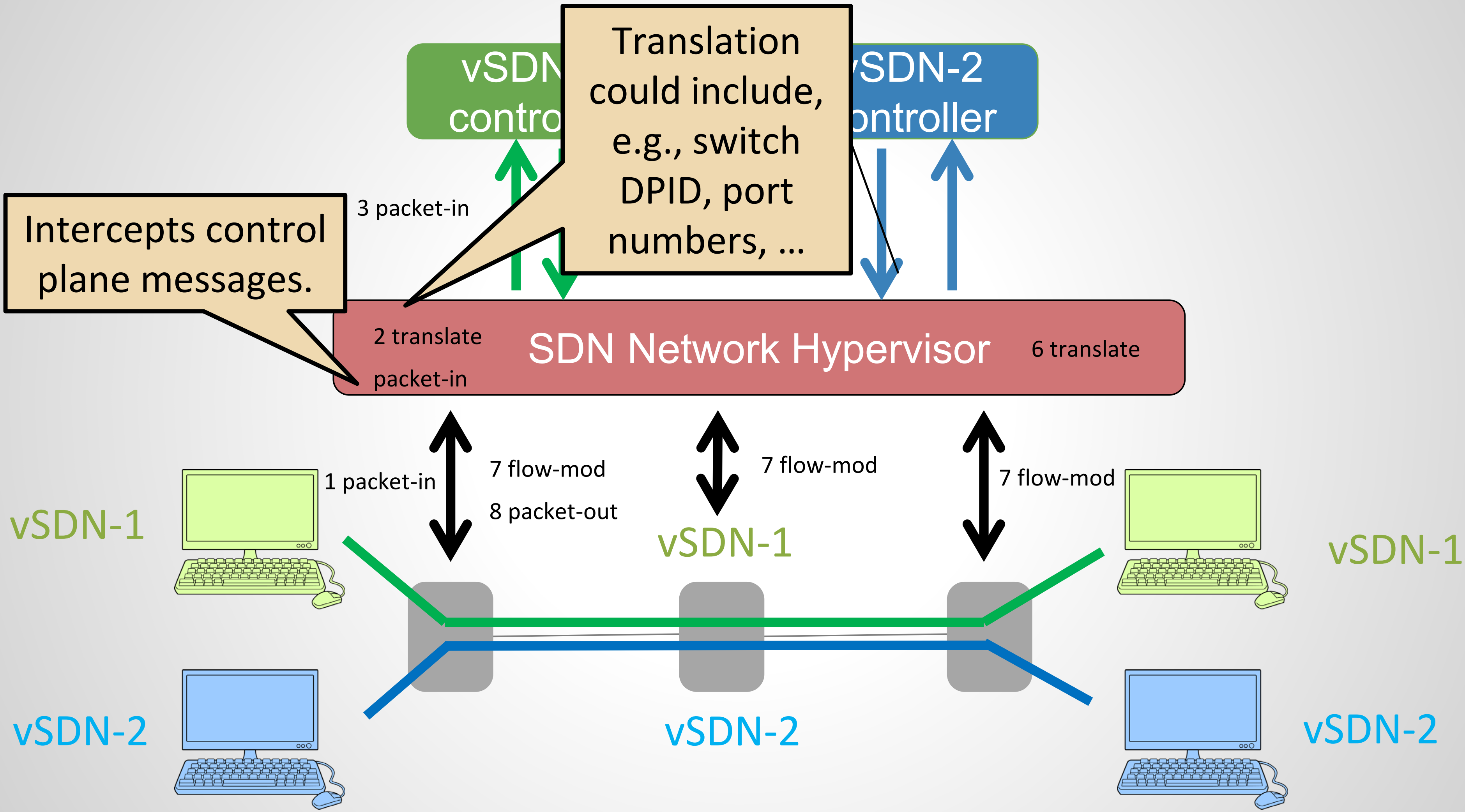
Realization: Virtual networks based on SDN

Models: Mind the Gap!



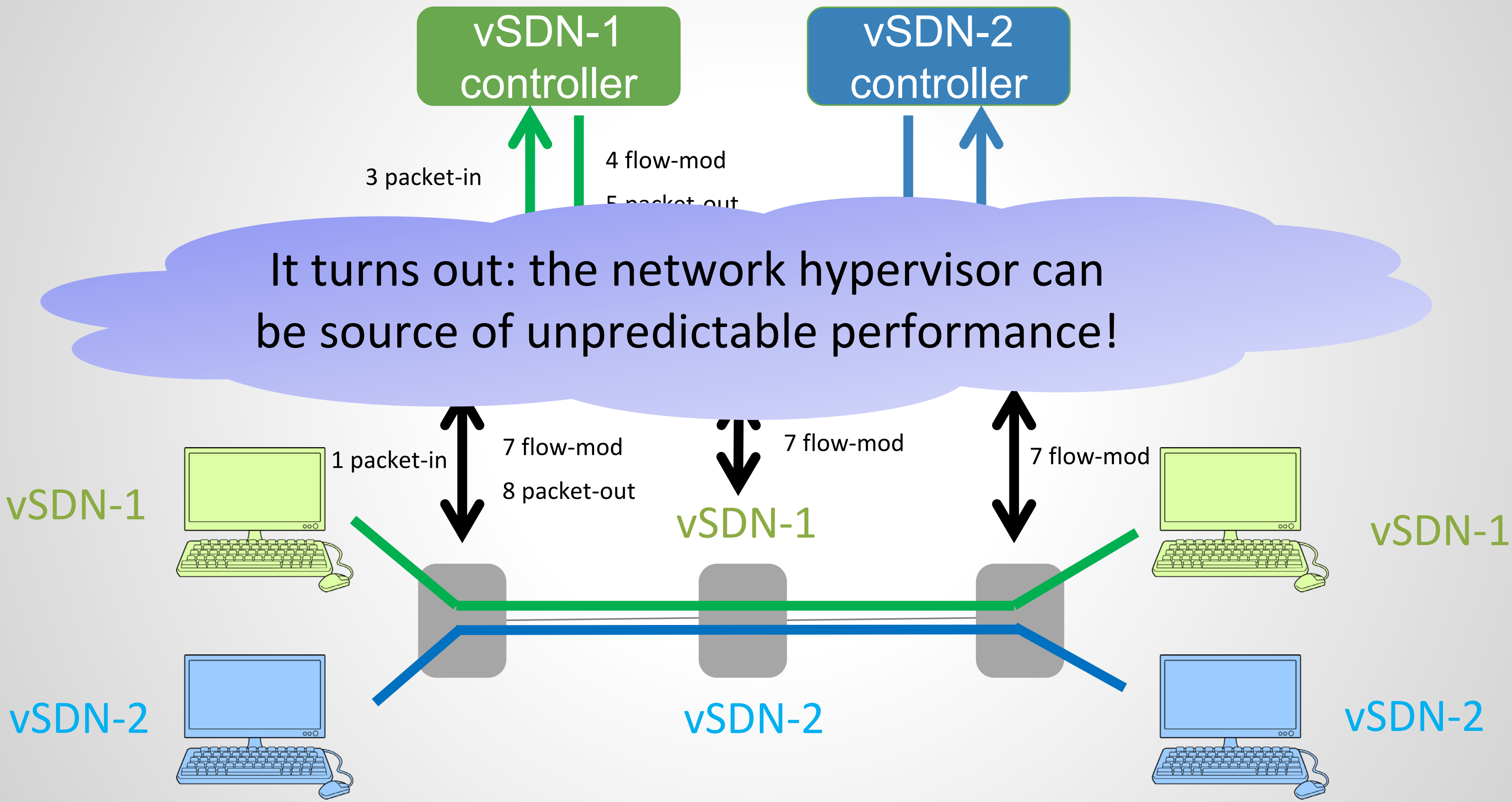
Realization: Virtual networks based on SDN

Models: Mind the Gap!



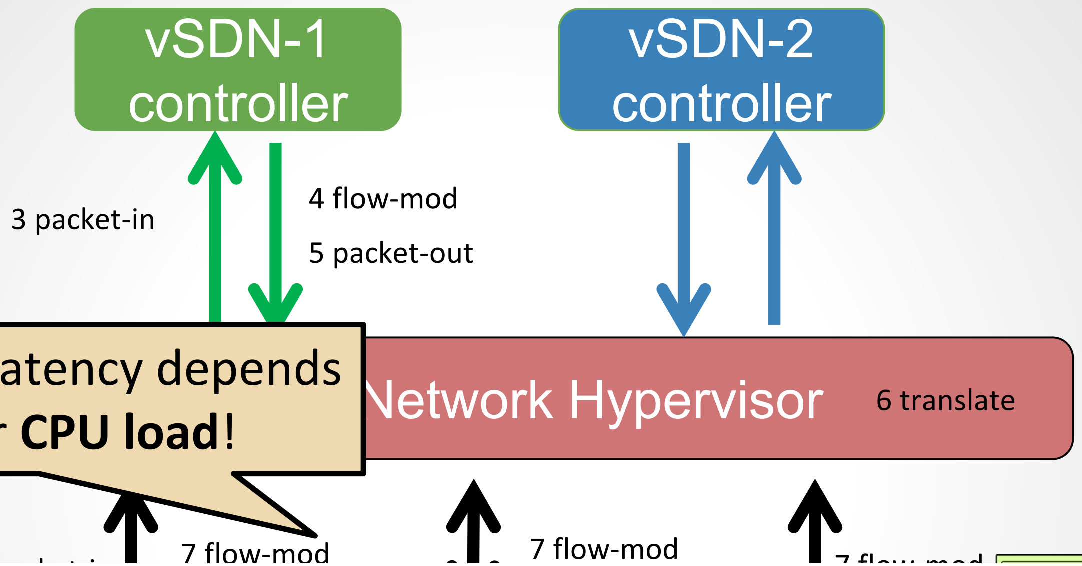
Realization: Virtual networks based on SDN

Models: Mind the Gap!

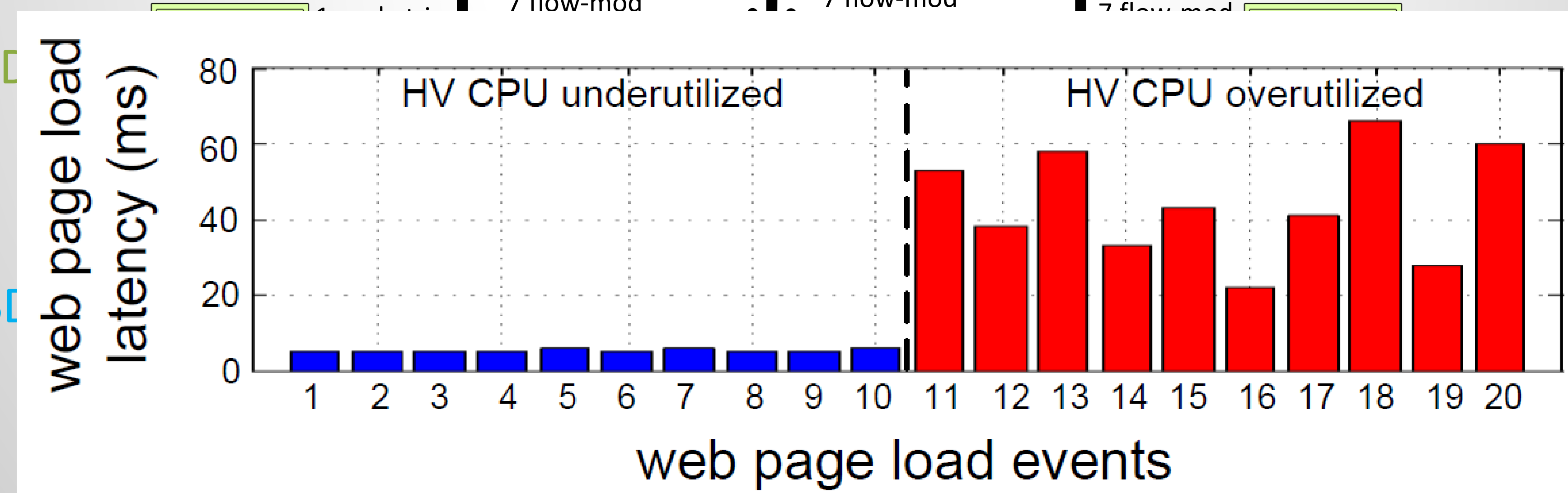


Realization: Virtual networks based on SDN

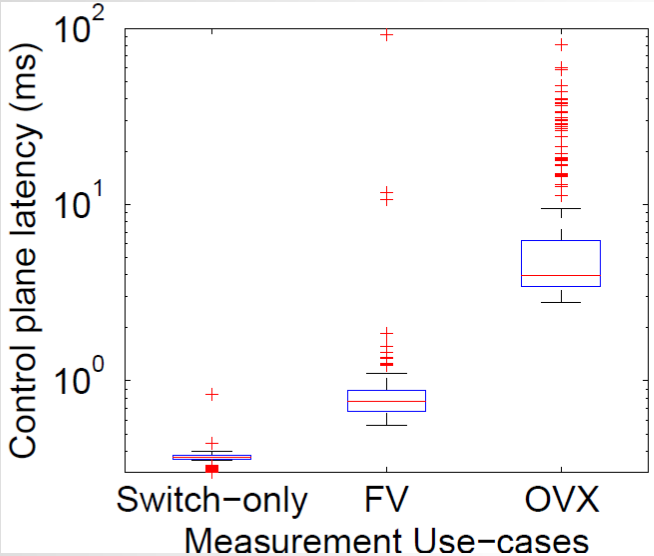
Models: Mind the Gap!



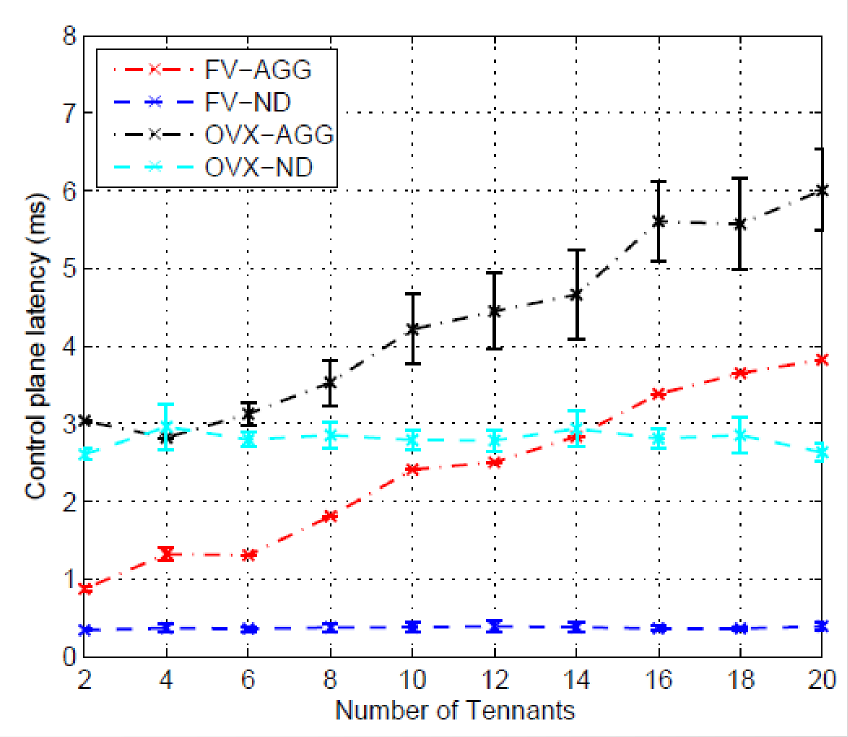
Experiment: web latency depends on hypervisor **CPU load**!



The Many Faces of Performance Interference



… number of tenants…



Performance also depends on hypervisor type…

(multithreaded or not, which version of Nagle's algorithm, etc.)

It's complex!

Invitation: A Roadtrip Through The Opportunities and Challenges of Network Isolation

- ❑ Opportunities
  - ❑ Algorithmic opportunities
  - ❑ Technological opportunities
- ❑ Challenges
  - ❑ Algorithmic challenges
  - ❑ Modelling challenges
  - ❑ **Security challenges**
- ❑ A perspective how AI can improve slicing efficiency and security

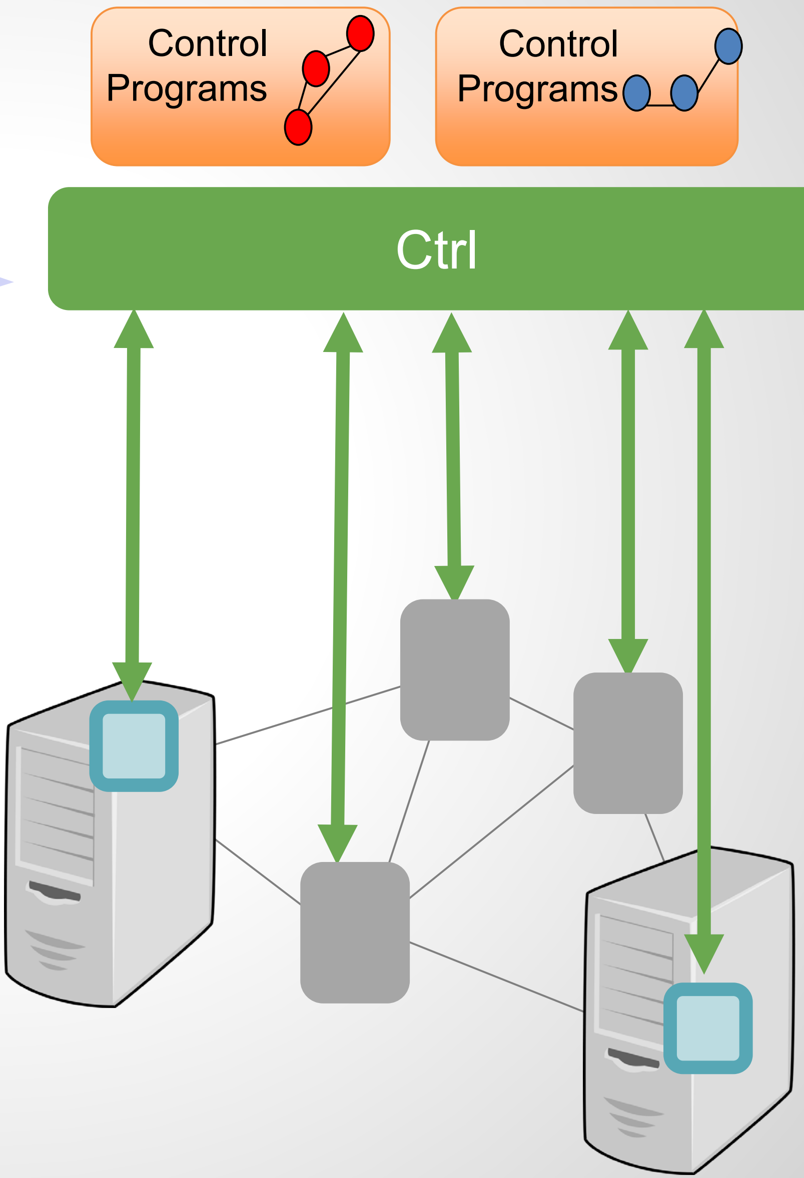
Road map 1927: Arizona and New Mexico

Programmable and Virtualized Networks

Increasingly centralized

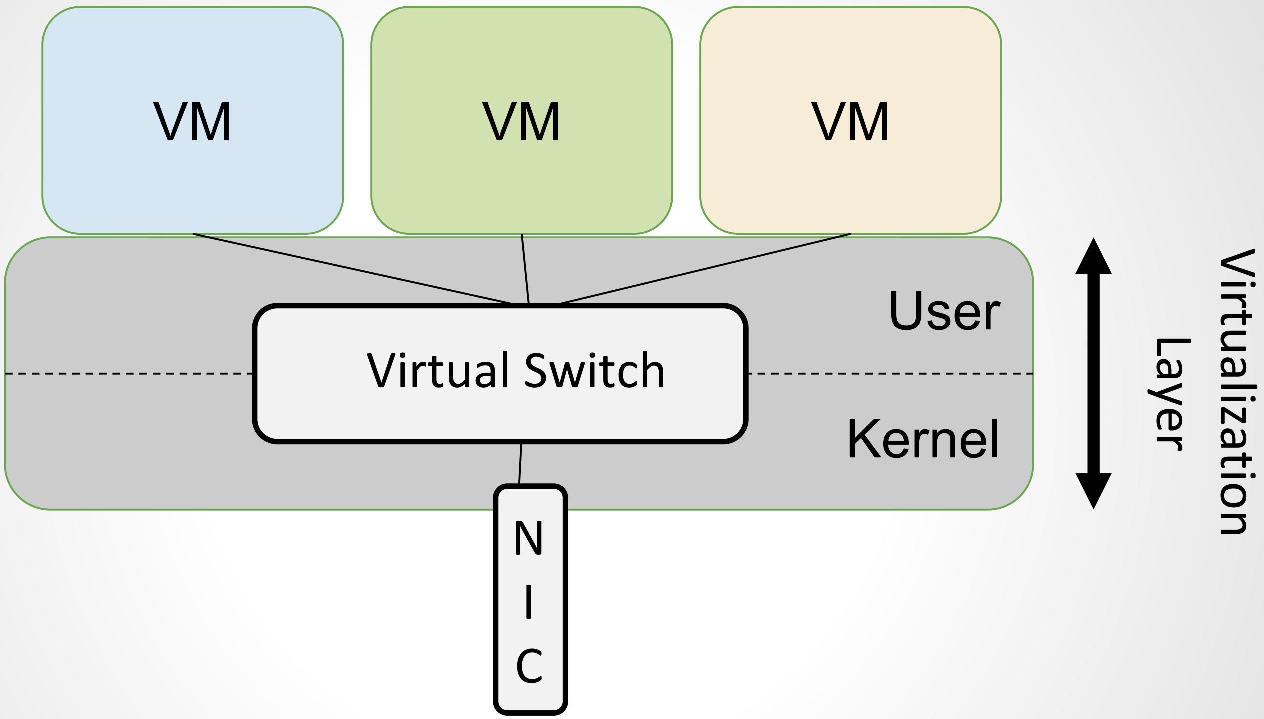
Networked systems:

Increasingly virtualized



Challenge: security!

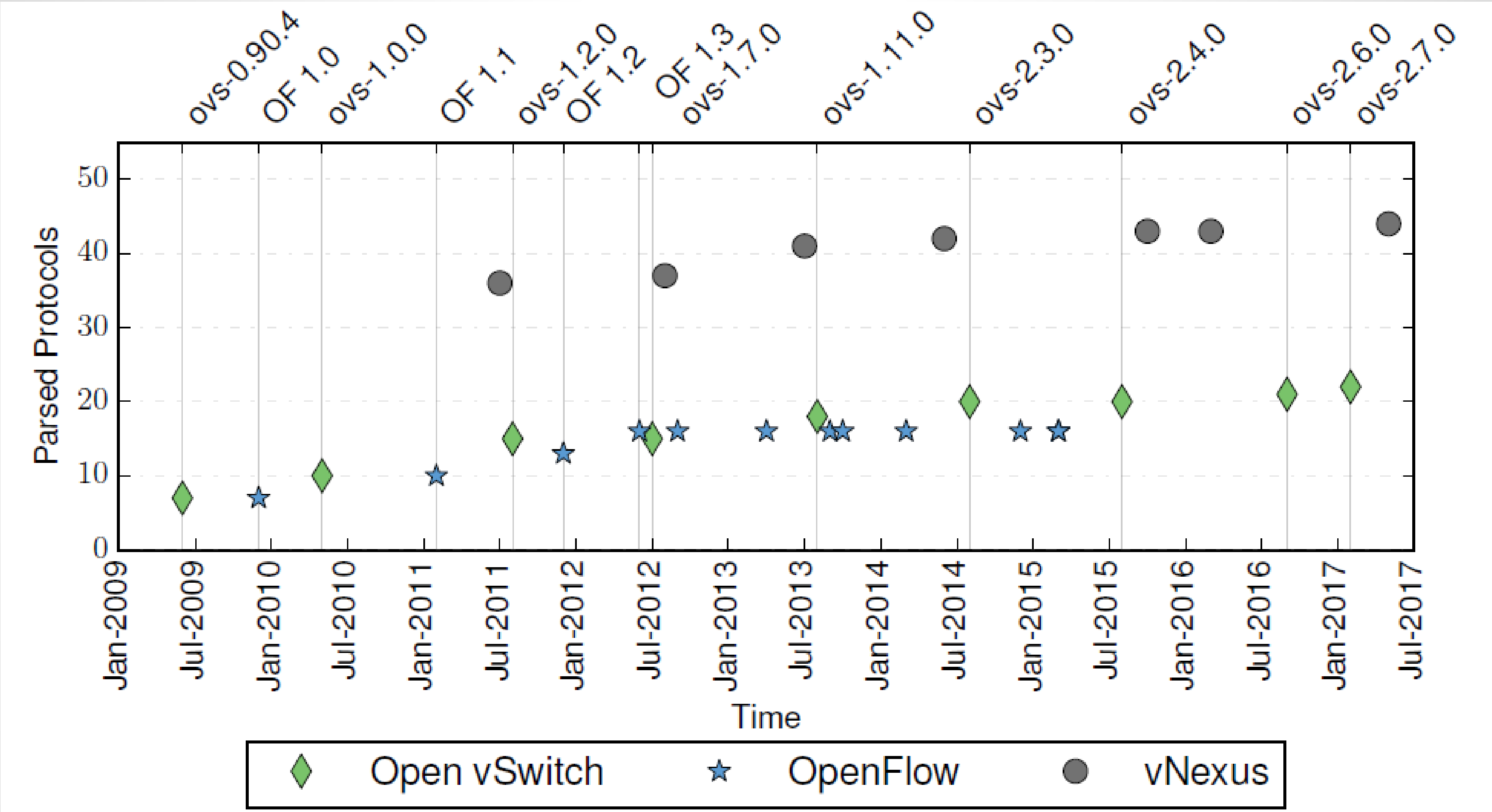
Potential New Attack Surface: Virtual Switches



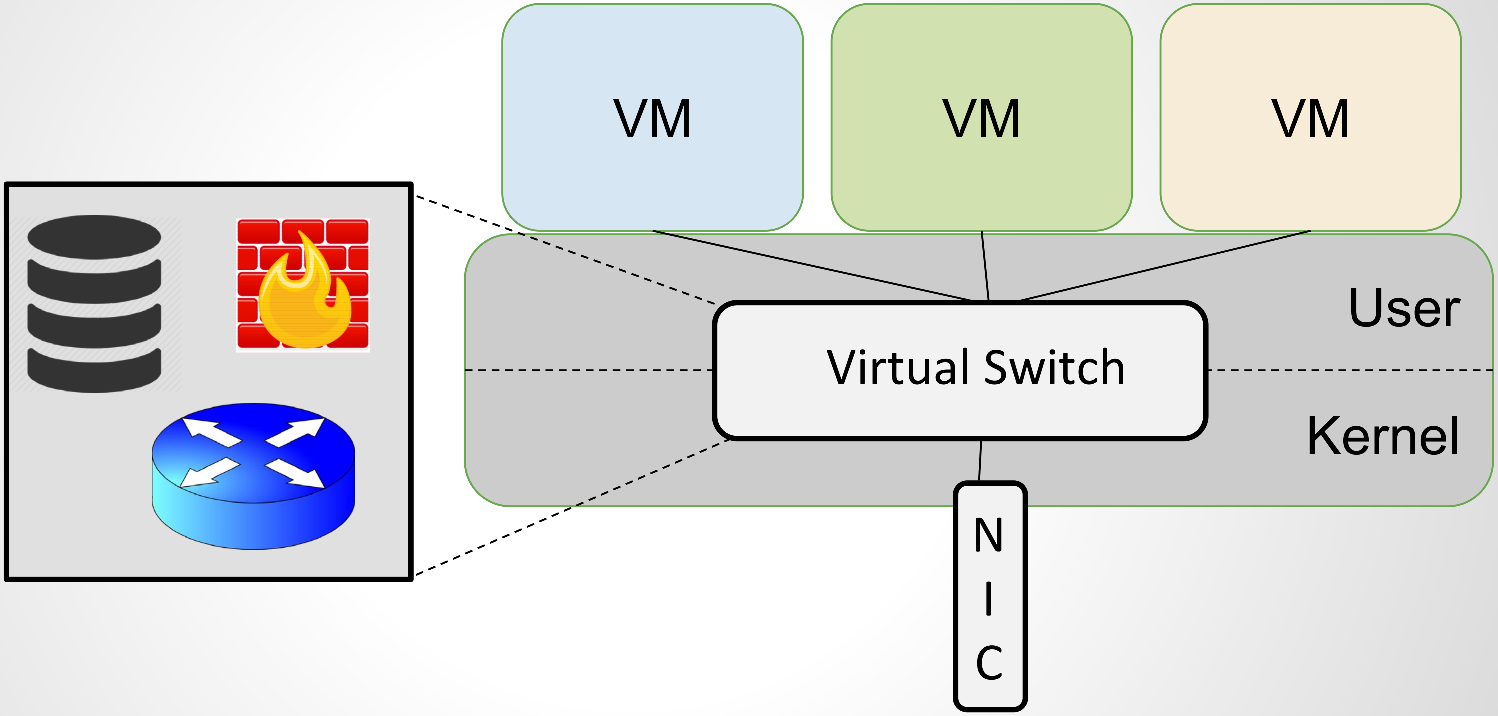
Virtual switches reside in the **server's virtualization layer** (e.g., Xen's Dom0). Goal: provide connectivity and isolation.

Increasing Complexity:
 # Parsed Protocols

Number of parsed high-level protocols constantly increases:



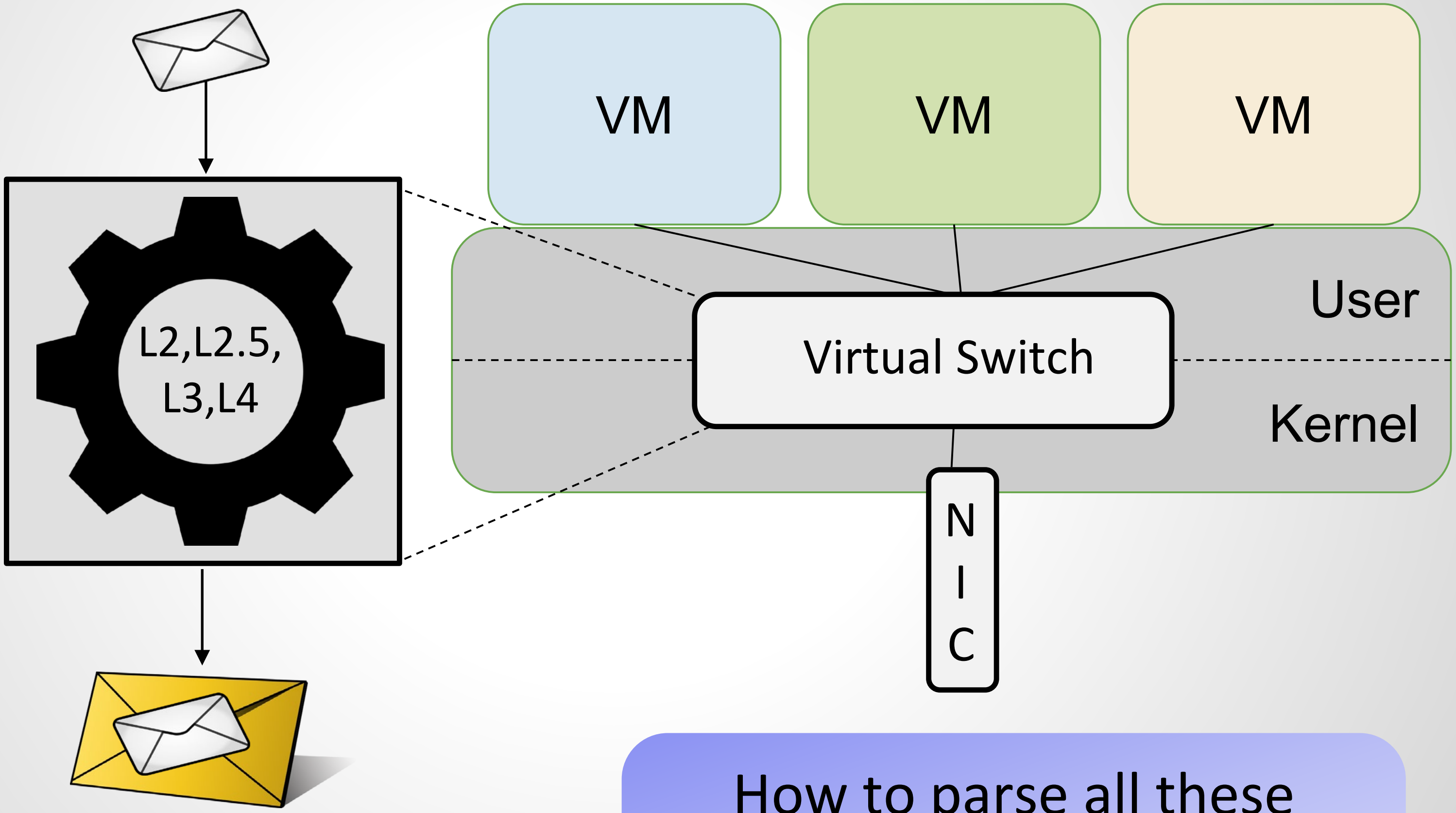
Increasing Complexity:
Introduction of middlebox functionality



Increasing workloads and advancements in network virtualization drive virtual switches to **implement middlebox functions** such as load-balancing, DPI, firewalls, etc.

Increasing Complexity: *Unified Packet Parsing*

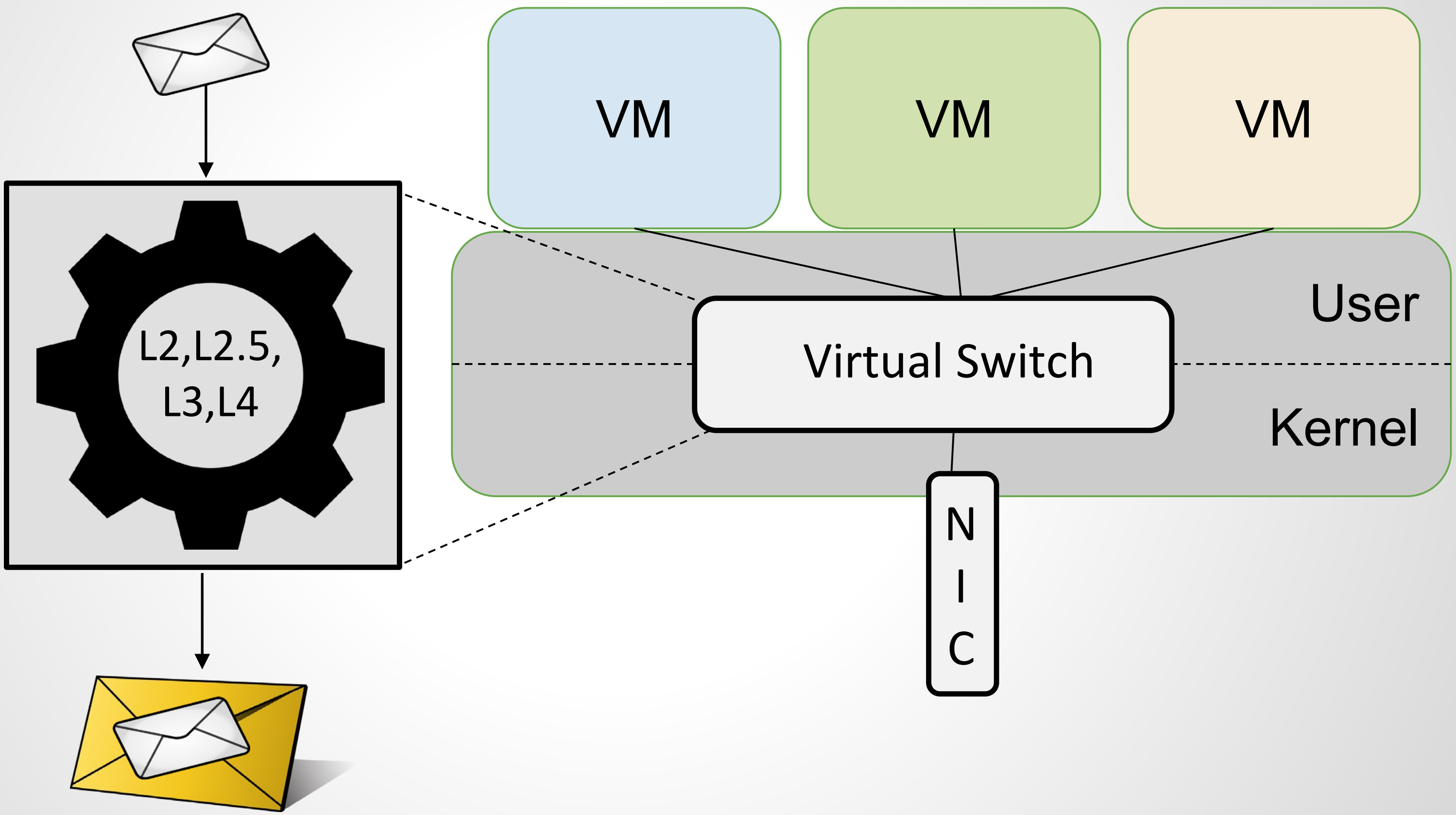
Ethernet
LLC
VLAN
MPLS
IPv4
ICMPv4
TCP
UDP
ARP
SCTP
IPv6
ICMPv6
IPv6 ND
GRE
LISP
VXLAN
PBB
IPv6 EXT HDR
TUNNEL-ID
IPv6 ND
IPv6 EXT HDR
IPv6HOPOPTS
IPv6ROUTING
IPv6Fragment
IPv6DESTOPT
IPv6ESP
IPv6 AH
RARP
IGMP



How to parse all these protocols without lowering forwarding performance?!

Increasing Complexity: *Unified Packet Parsing*

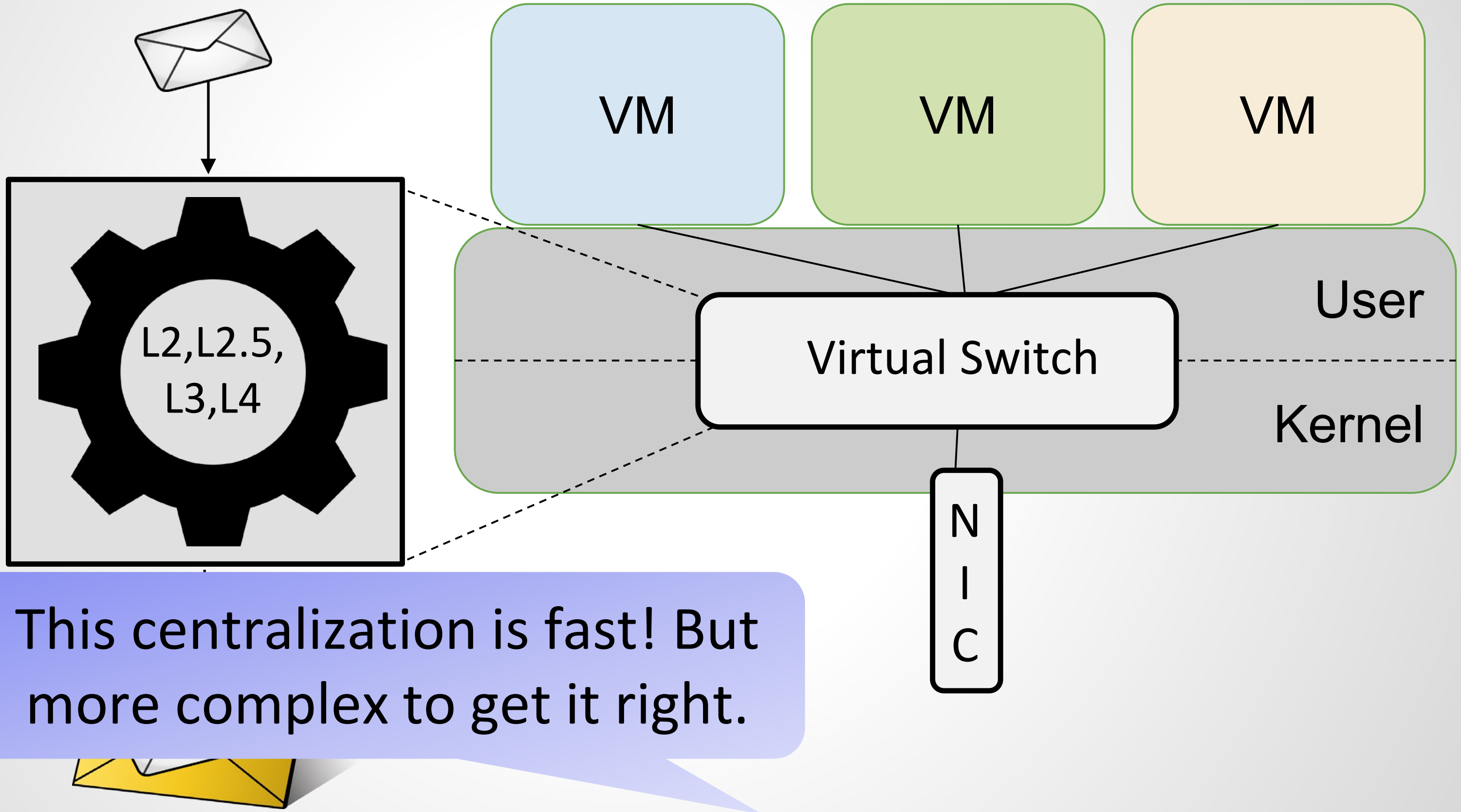
Ethernet
LLC
VLAN
MPLS
IPv4
ICMPv4
TCP
UDP
ARP
SCTP
IPv6
ICMPv6
IPv6 ND
GRE
LISP
VXLAN
PBB
IPv6 EXT HDR
TUNNEL-ID
IPv6 ND
IPv6 EXT HDR
IPv6HOPOPTS
IPv6ROUTING
IPv6Fragment
IPv6DESTOPT
IPv6ESP
IPv6 AH
RARP
IGMP



Unified packet parsing allows parse more and more protocols efficiently: in a single pass!

Increasing Complexity: *Unified Packet Parsing*

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



This centralization is fast! But more complex to get it right.

Unified packet parsing allows parse more and more protocols efficiently: in a single pass!

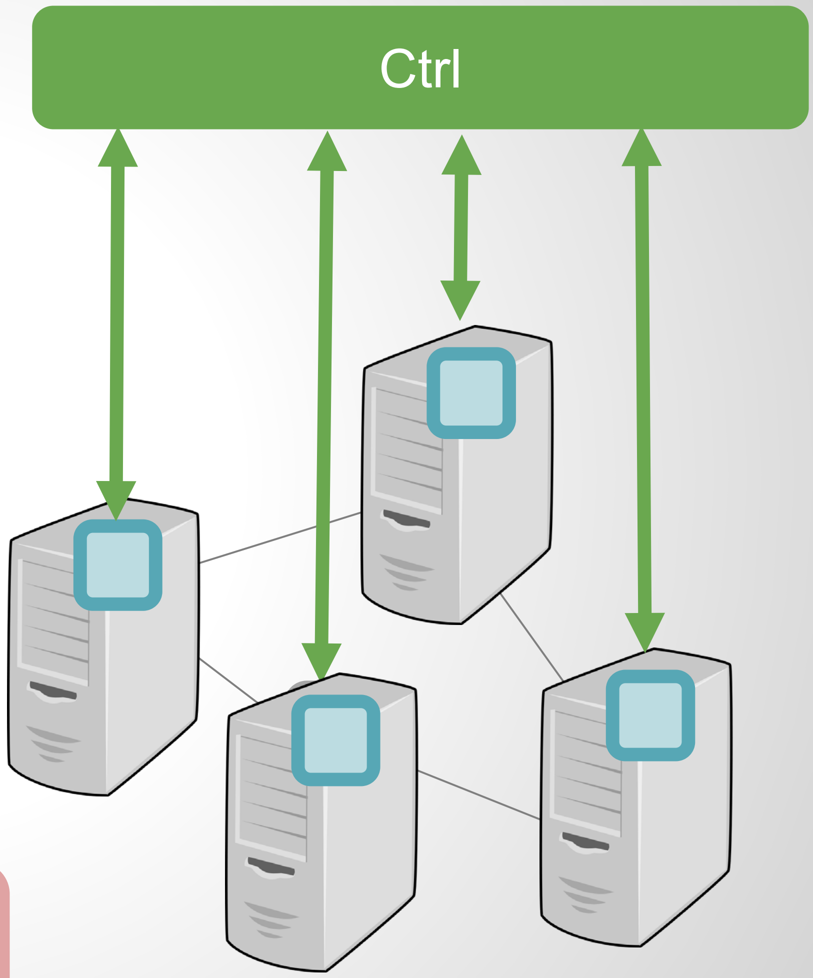
Complexity: The Enemy of Security!

- Data plane security not well-explored (in general, not only virtualized): most security research on control plane

- Two conjectures:

1. Virtual switches increase the attack surface.

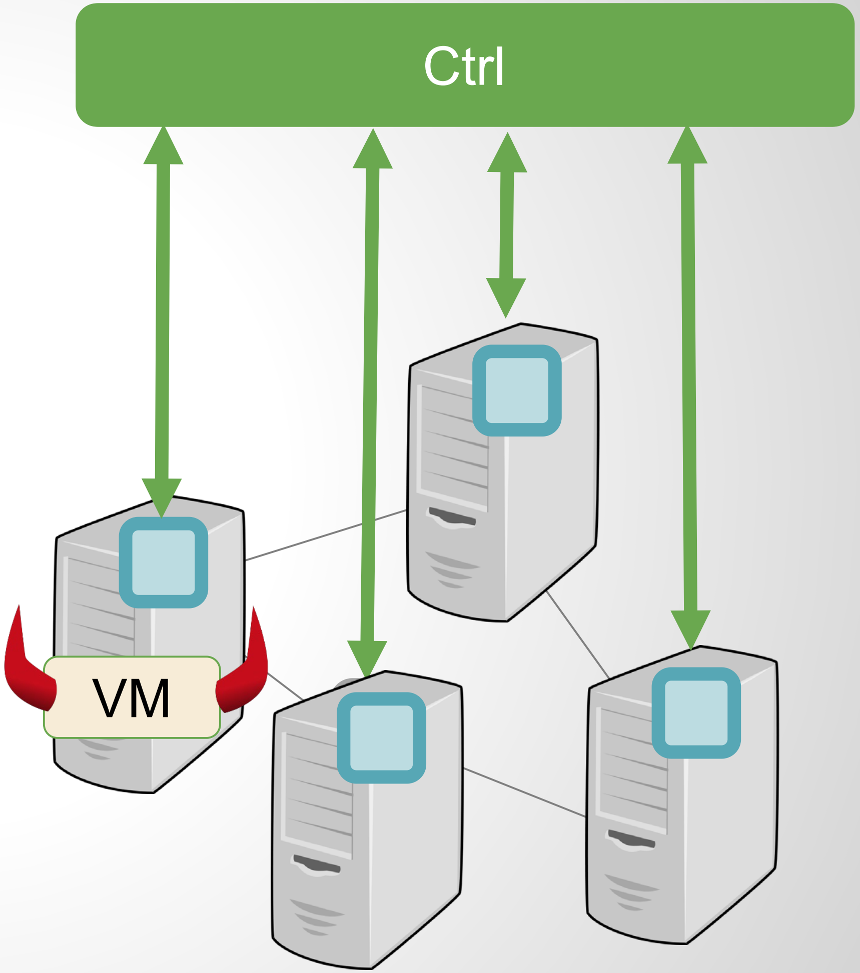
2. Impact of attack larger than with traditional data planes.



The Attack Surface: Closer...

Attack surface becomes closer:

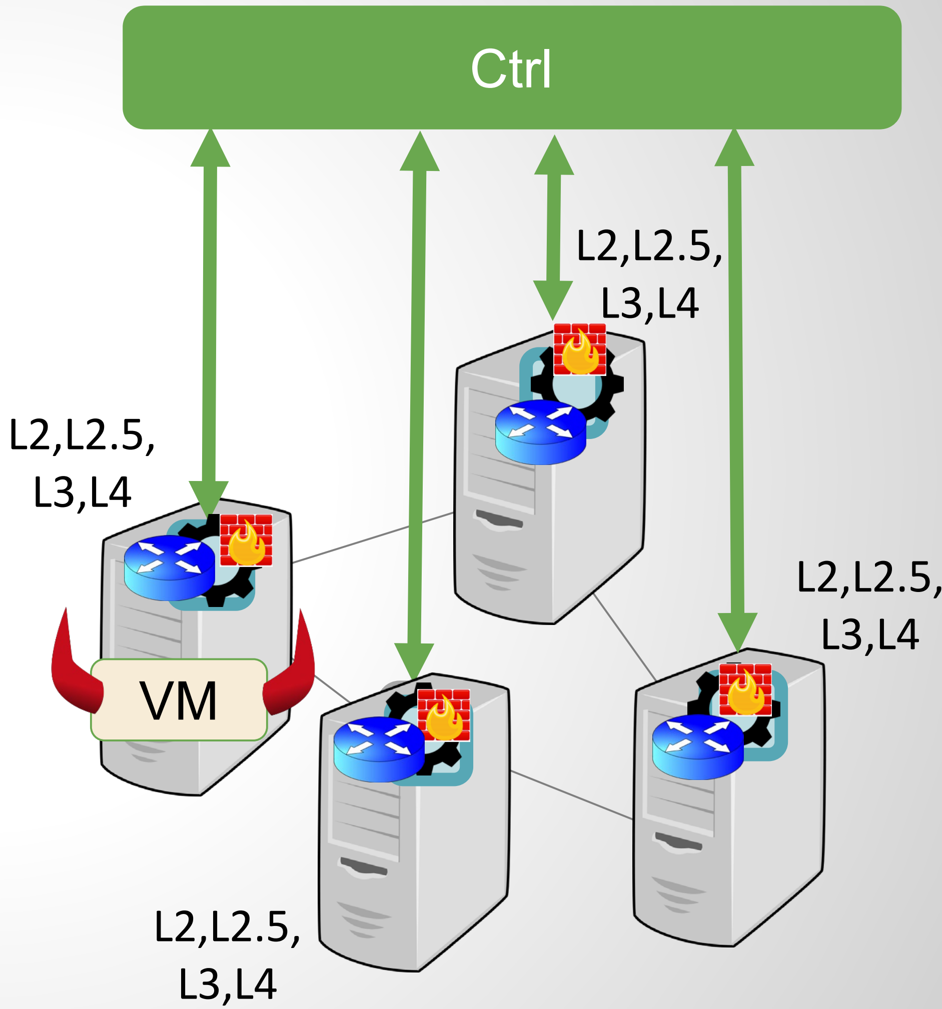
- ❑ Packet parser typically integrated into the code base of virtual switch
- ❑ First component of the virtual switch to process network packets it receives from the network interface
- ❑ May process attacker-controlled packets!



The Attack Surface: … More Complex …

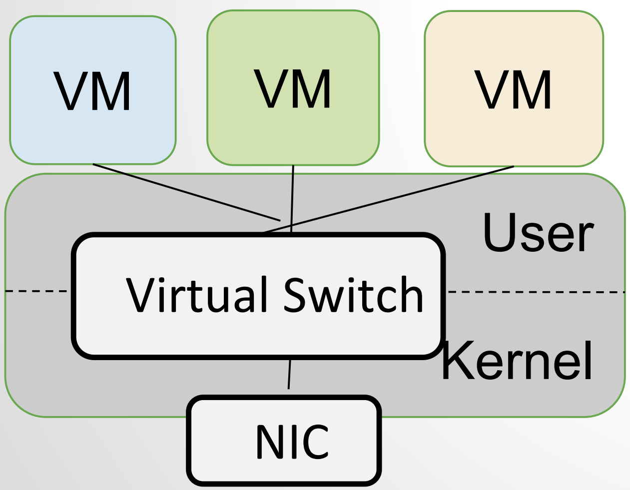
Ethernet
LLC
VLAN
MPLS
IPv4
ICMPv4
TCP
UDP
ARP
SCTP
IPv6
ICMPv6
IPv6 ND
GRE
LISP
VXLAN

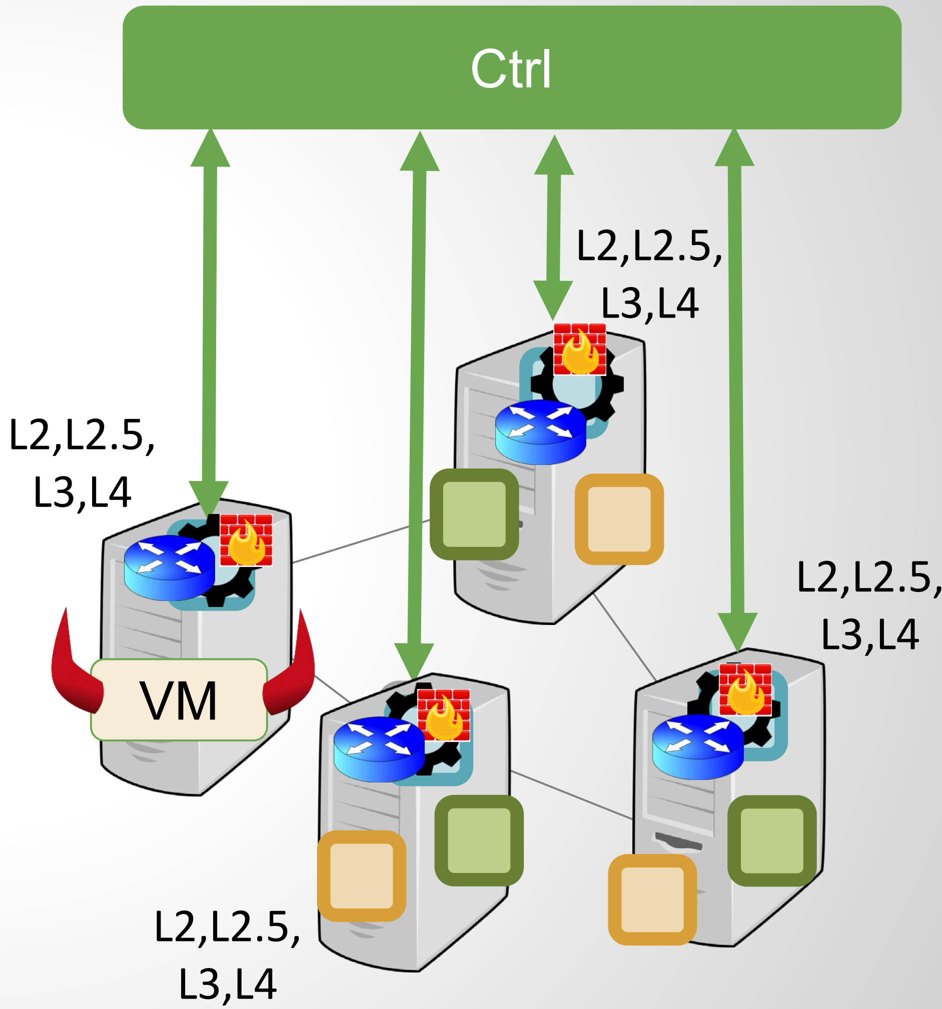
PBB
IPv6 EXT HDR
TUNNEL-ID
IPv6 ND
IPv6 EXT HDR
IPv6HOPOPTS
IPv6ROUTING
IPv6Fragment
IPv6DESTOPT
IPv6ESP
IPv6 AH
RARP
IGMP



… Elevated Priviledges and Collocation …

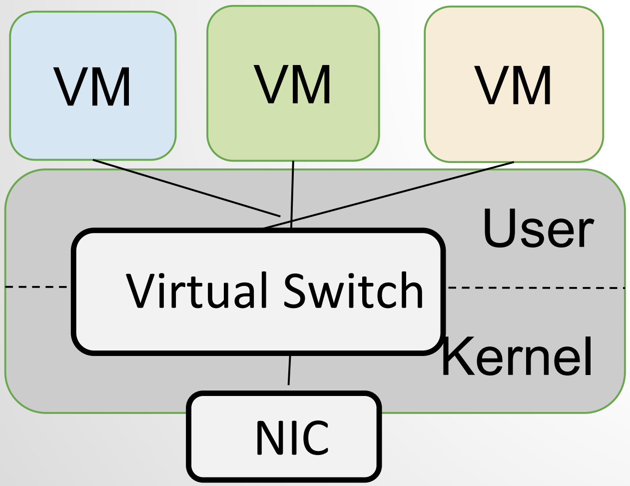
- Collocated (at least partially) with hypervisor's Dom0 kernel space, guest VMs, image management, block storage, identity management, …

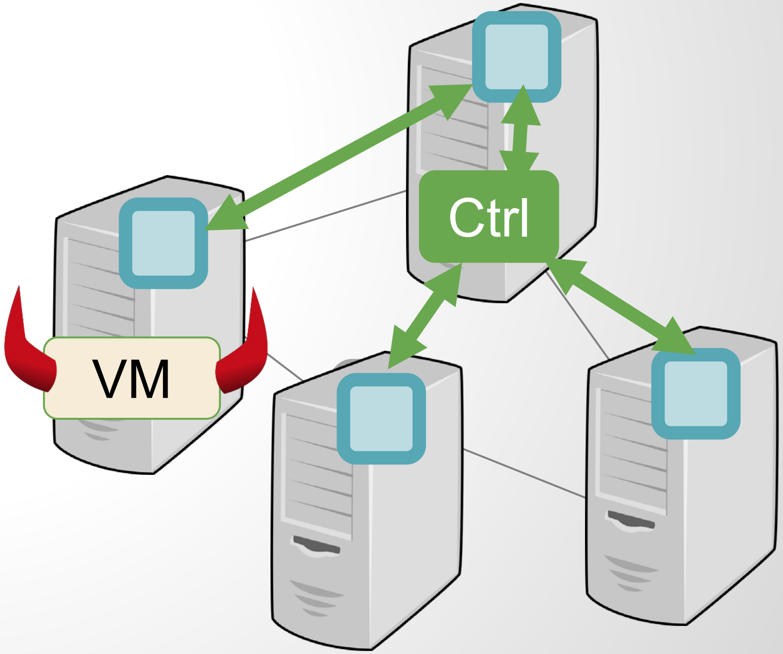




… Elevated Priviledges and Collocation …

- Collocated (at least partially) with hypervisor's Dom0 kernel space, guest VMs, image management, block storage, identity management, …



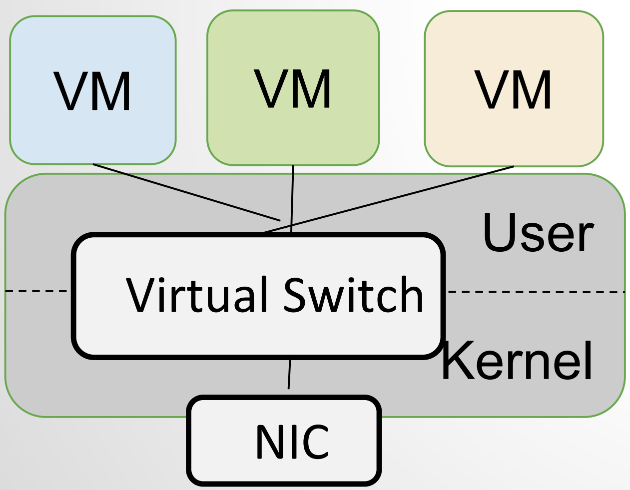


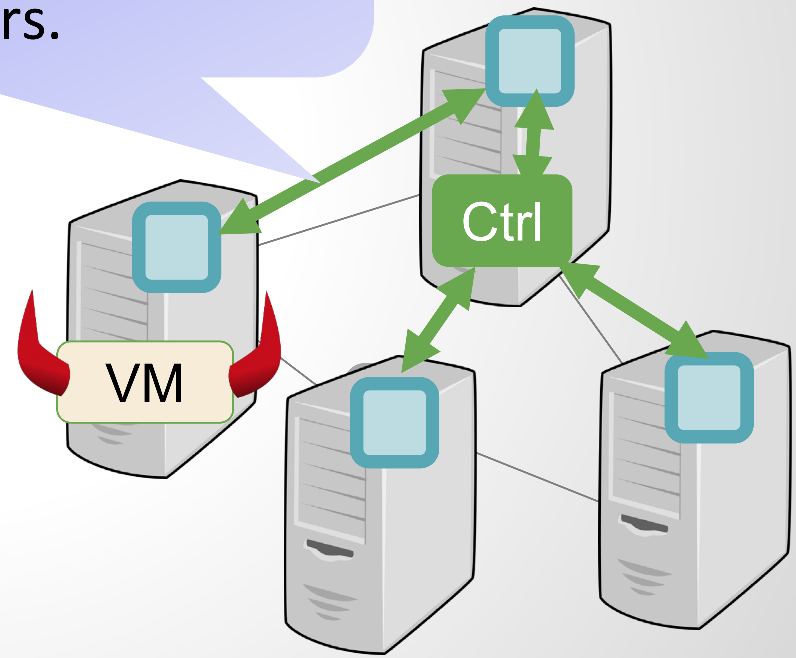
- … the controller itself.

… Centralization …

- Collocate
with hype
space), g
managem
identity m

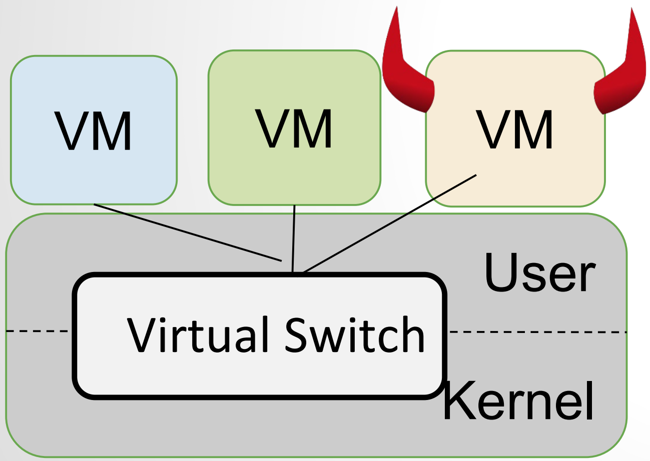
Available communication channels to (SDN/Openstack) controller! Controller needs to be reachable from all servers.





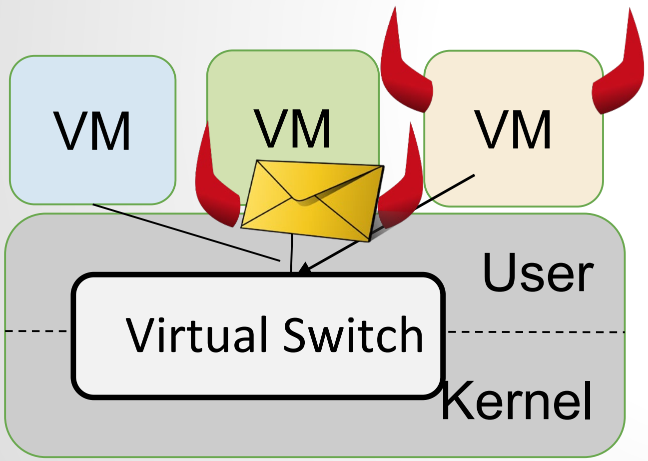
- … the controller itself.

Larger Impact: Case Study OVS



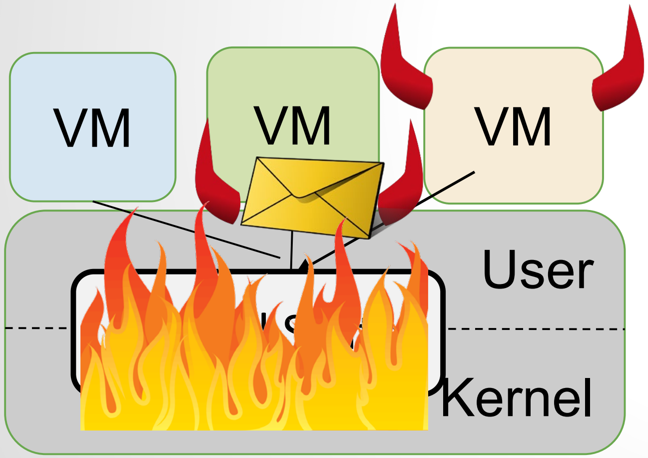
1. Rent a VM in the cloud (**cheap**)

Larger Impact: Case Study OVS



2. Send **malformed MPLS packet** to virtual switch (**unified parser** parses label stack packet beyond the threshold)

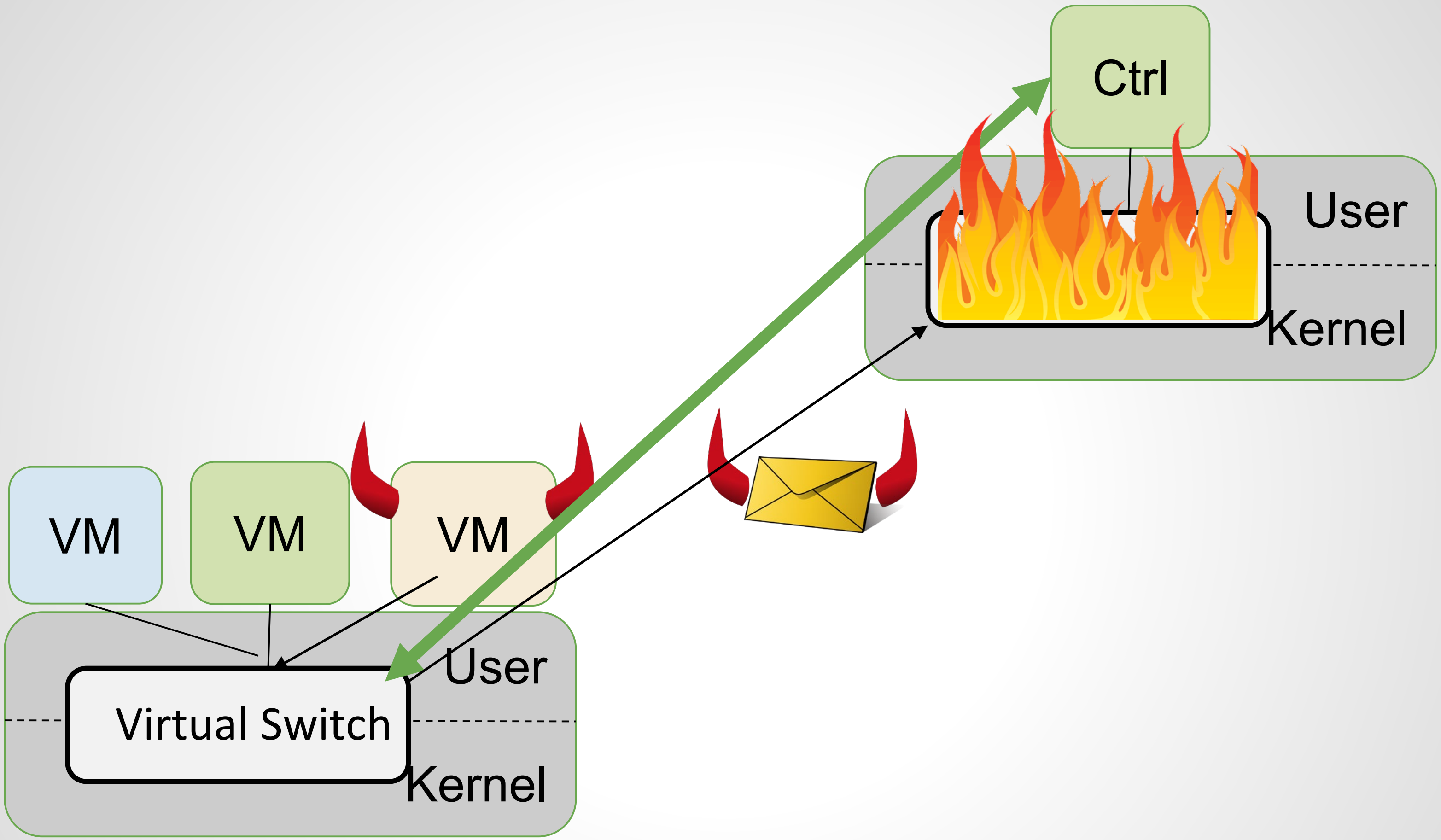
Larger Impact: Case Study OVS



3. **Stack buffer overflow** in (unified) MPLS parsing code:

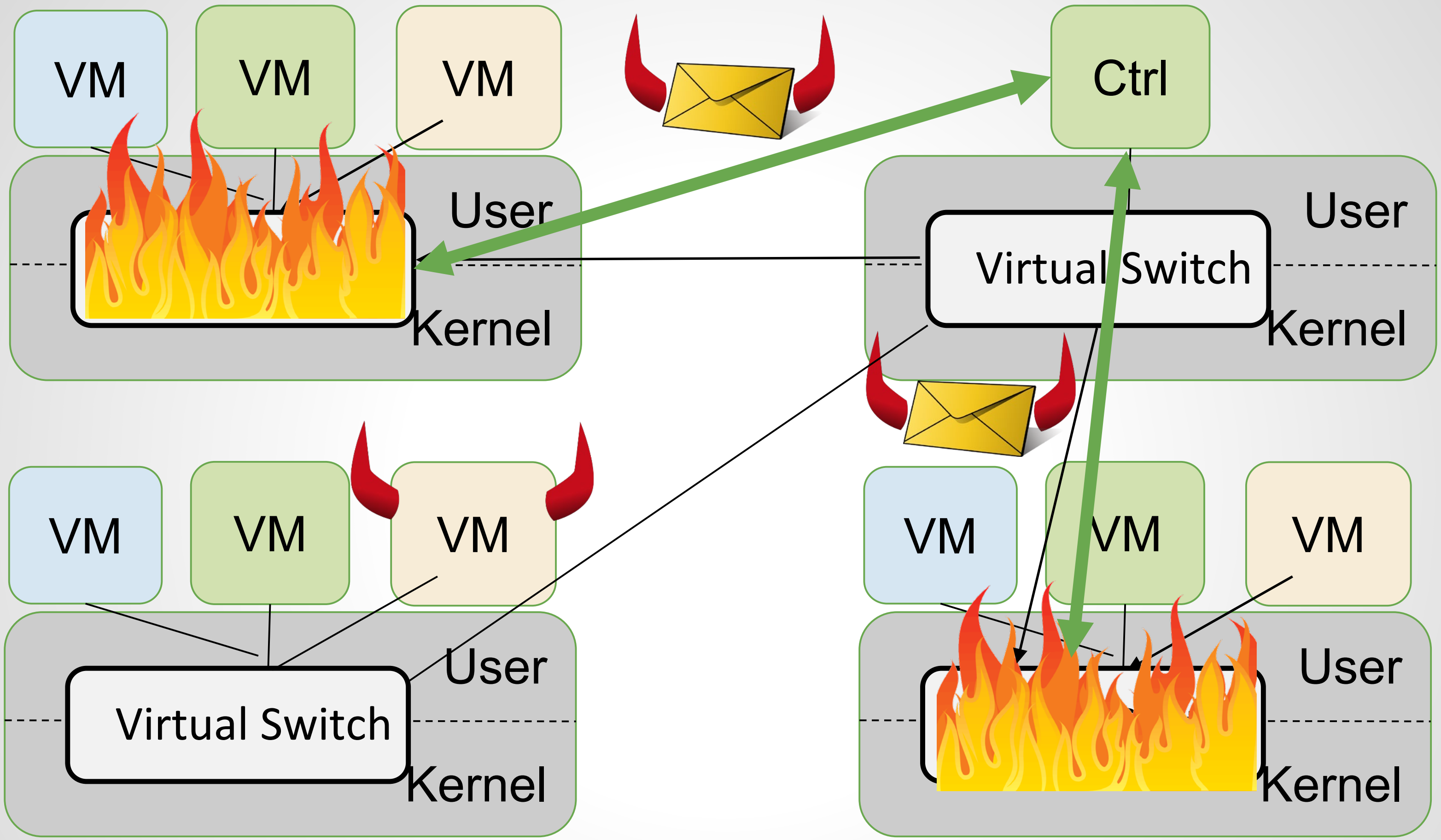
enables remote code execution

Larger Impact: Case Study OVS



4. Send malformed packet to server (virtual switch) where controller is located (use existing communication channel)

Larger Impact: Case Study OVS



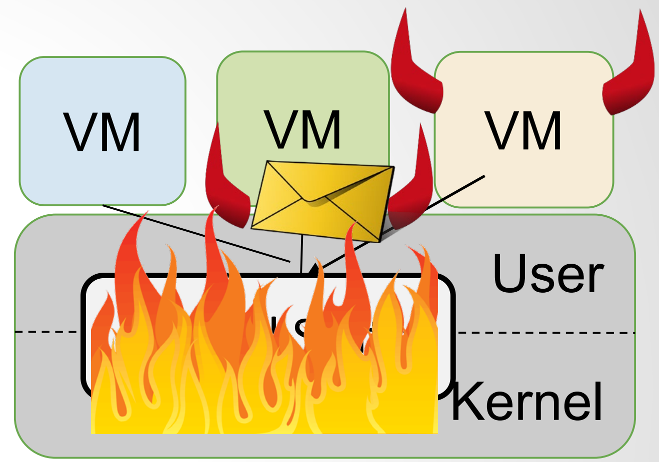
5. Spread

A New Threat Model

- ❑ Limited skills required
  - ❑ Use standard fuzzer to find crashes
  - ❑ Construct malformed packet
  - ❑ Build ROP chain

- ❑ Limited resources
  - ❑ rent a VM in the cloud

- ❑ No physical access needed



No need to be a state-level attacker to compromise the dataplane (and beyond)!

Similar problems in NFV: need even more complex parsing/processing. And are often built on top of OvS.

Invitation: A Roadtrip Through The Opportunities and Challenges of Network Isolation

- ❑ Opportunities
  - ❑ Algorithmic opportunities
  - ❑ Technological opportunities

- ❑ Challenges
  - ❑ Modelling challenges
  - ❑ Security challenges

- ❑ **A perspective how AI can improve slicing efficiency and security**

Road map 1927: Arizona and New Mexico

A Case for AI?

1. Modelling virtualized and softwarized systems is complex

- ❑ E.g., recall our SDN setup
- ❑ Often, many algorithms and parameters involved
- ❑ Wireless/radio components likely to increase complexity

2. In practice, `optimal' resource sharing typically achieved with statistical multiplexing

- ❑ Requires data: the more the better the statistics and hence the efficiency

3. Resource allocation algorithms are often executed repeatedly

- ❑ E.g., routing, embedding, switching...

A Case for AI?

1. Modelling virtualized and softwarized systems is complex

- ❑ E.g., recall our SDN setup
- ❑ Often, many algorithms and parameters involved
- ❑ Wireless/radio components likely to increase complexity

A case for AI?

2. In practice, `optimal' resource sharing typically achieved with statistical multiplexing

- ❑ Requires data: the more the better the statistics and hence the efficiency

A case for AI?

3. Resource allocation algorithms are often executed repeatedly

- ❑ E.g., routing, embedding, switching...

A case for AI?

A Case for AI?

1. Modelling virtualized and softwarized systems is complex

A case for AI?

- E.g., recall our SDN setup
- Often, many algorithms and parameters involved
- Wireless/radio components likely to increase complexity

2. In practice, `optimal' resource sharing typically achieved with statistical multiplexing

A case for AI?

- Requires data: the more the better the statistics and hence the efficiency

3. Resource allocation algorithms are often executed repeatedly

A case for AI?

- E.g., routing, embedding, switching...

Today's Approach to Operate Networks

Monitors → Network Problem

Optimizes → Solution

Designs → Performance Evaluation

With more complex networks: need for automation!

What Self-Driving Networks Could Do?

Self-Monitoring → Network Problem

Self-Optimizing → Solution

E.g. O’zapf (BIGDAMA’17)

Self-Benchmarking → Performance Evaluation

E.g, NetBOA (NetAI’19)

What Self-Driving Networks Could Do?

Self-Monitoring → Network Problem

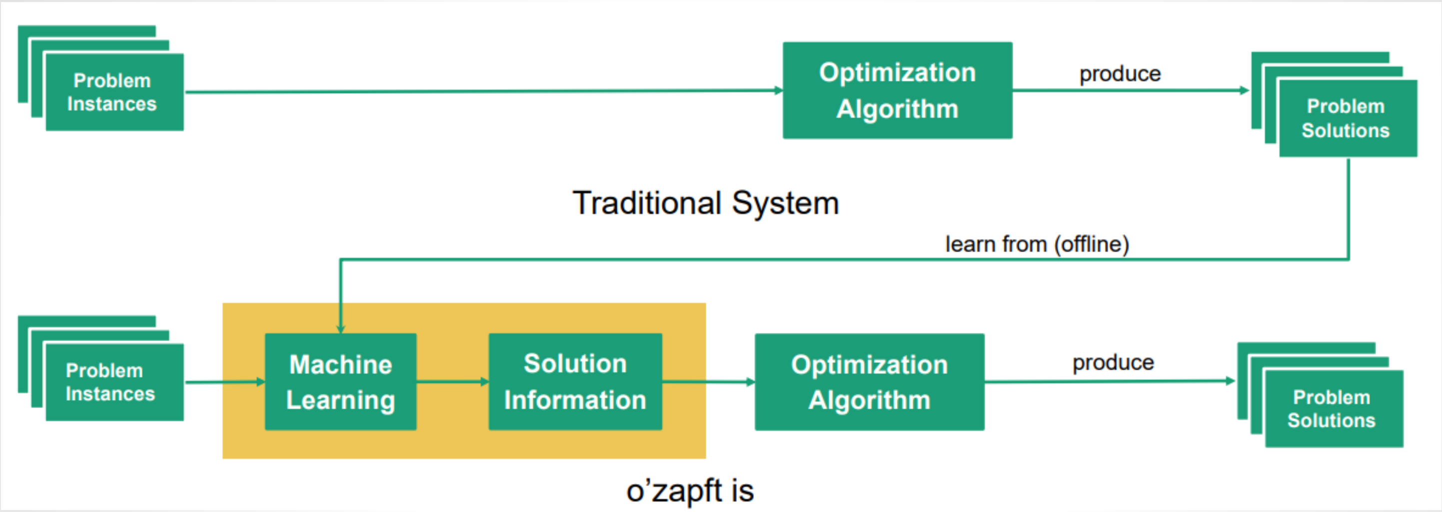
Self-Optimizing → Solution

E.g. O'zapf (BIGDAMA'17)

Self-Benchmarking → Performance Evaluation

E.g, NetBOA (NetAI'19)

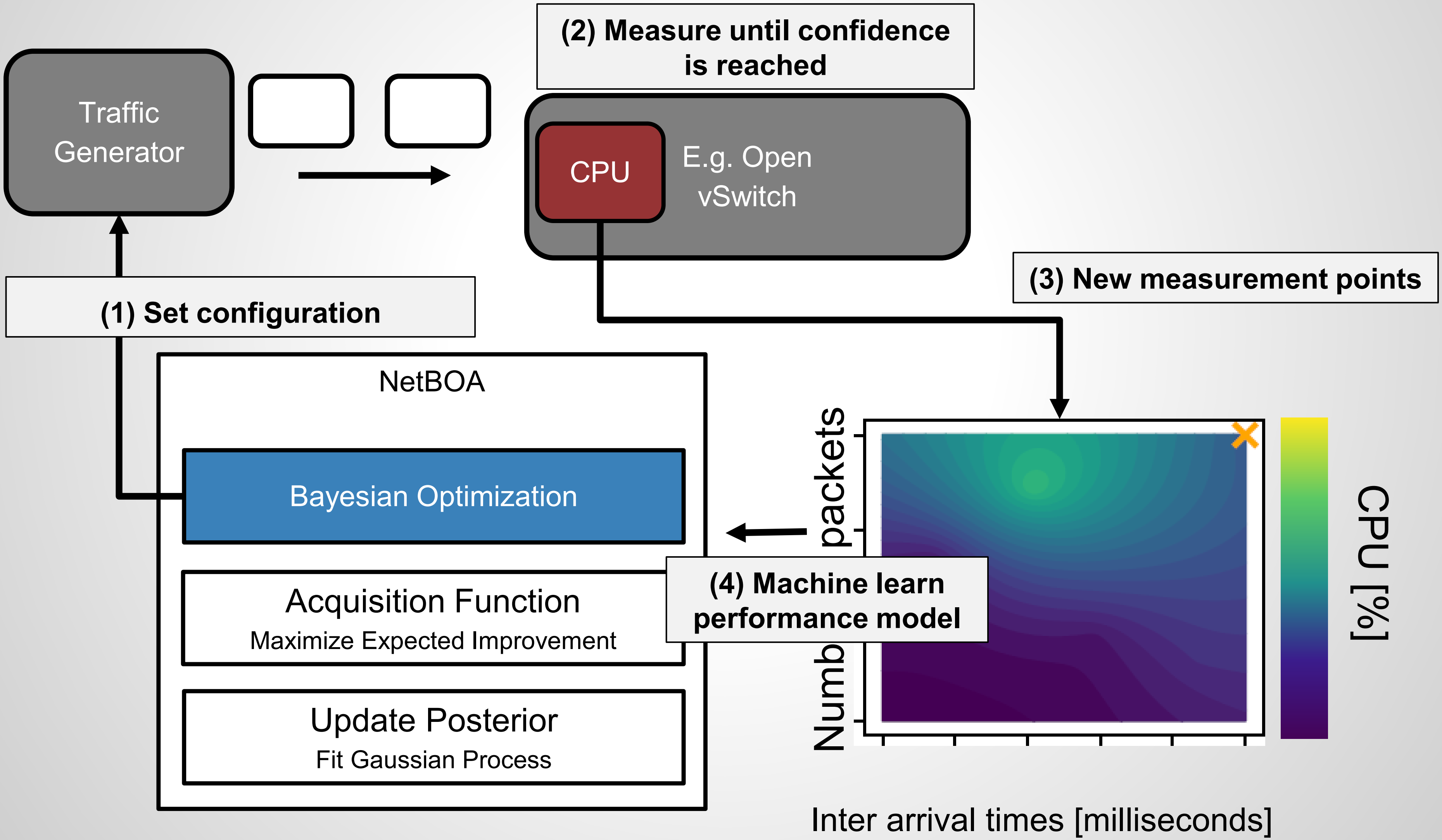
Example: Data-Driven Algorithms



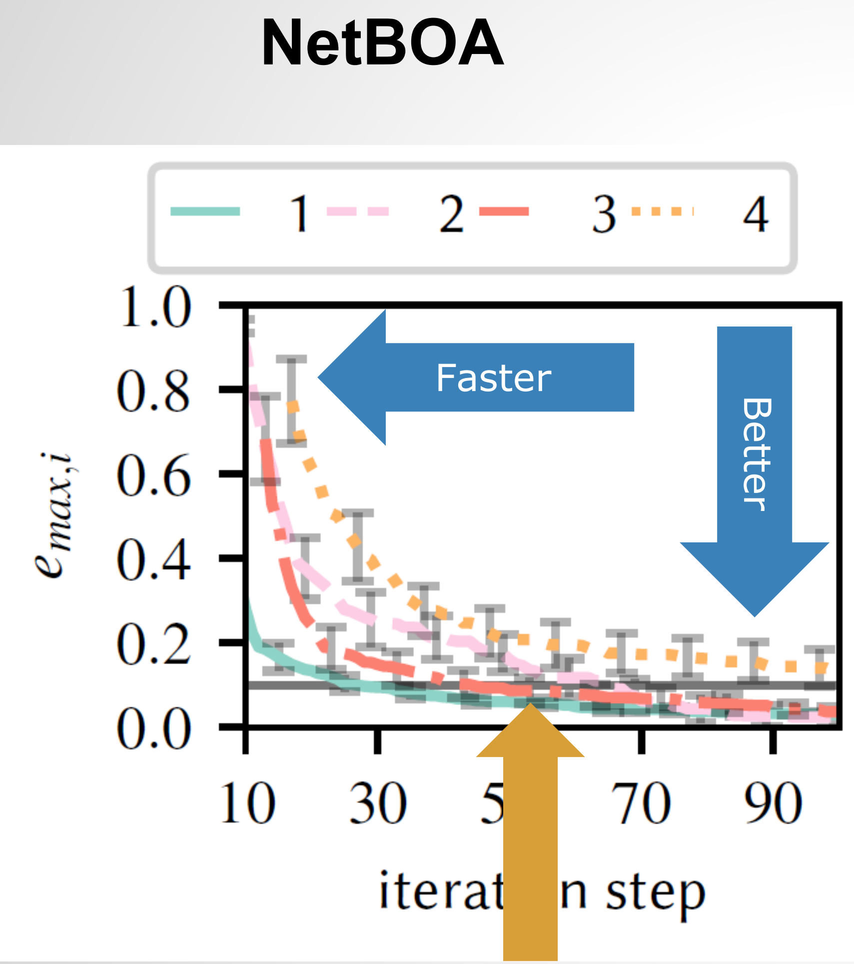
Can we learn from past solutions?

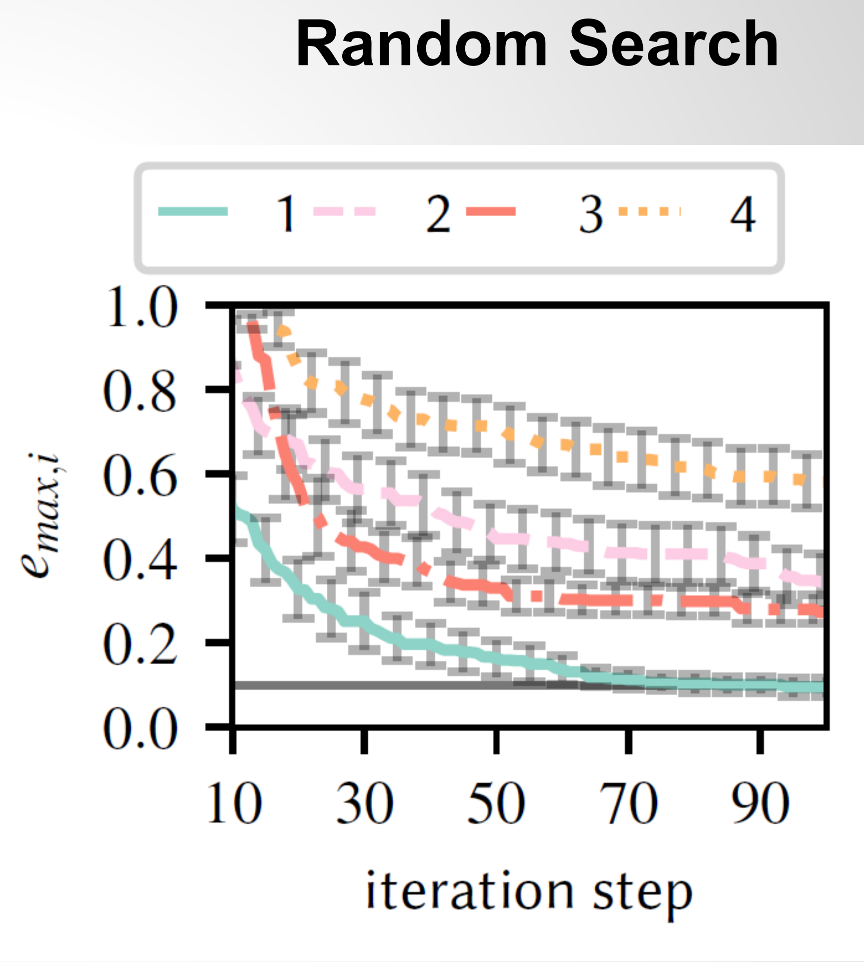
- E.g., to speed up future solutions?

Example: NetBOA
Automated Learning of “Bad Inputs”



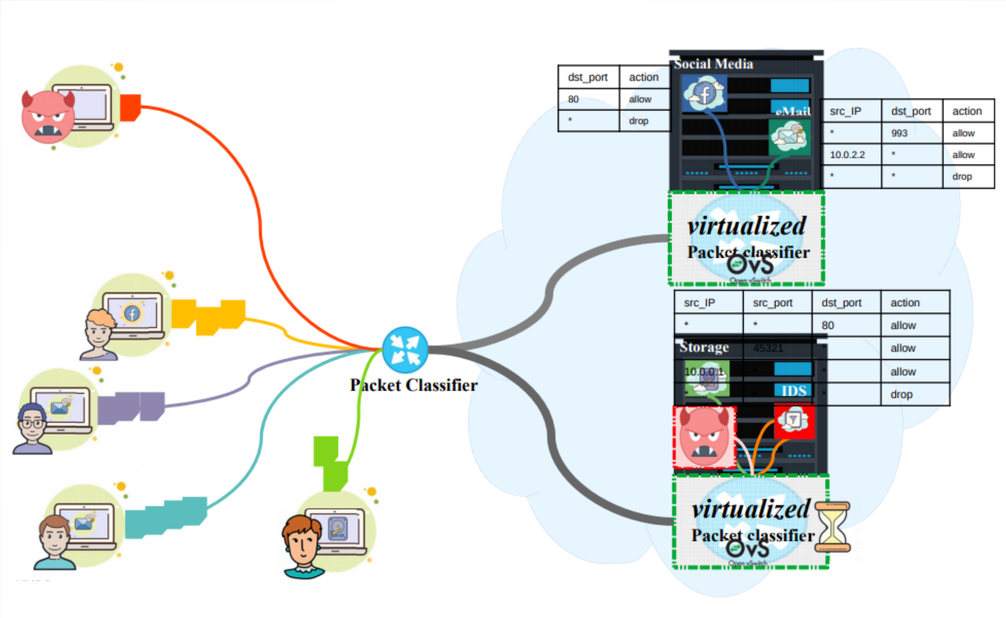
NetBOA vs Random Search





24 % higher CPU utilization

May Also Be Exploited: Algorithmic Complexity Attacks

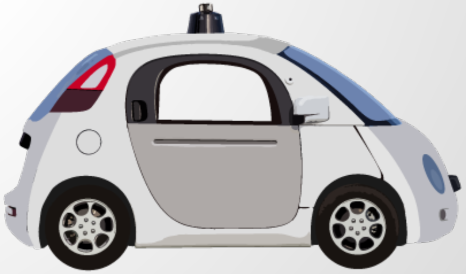


E.g., automated learning of bad inputs to packet classifier

- E.g., difficult regular expressions
- Severely affects performance of OvS
- Can result in denial-of-service

Challenges of AI-Based and Self-Driving Networks

- How much control are we willing to give away?
- Can a self-* network realize its **limits**?
- E.g., when quality of **input data** is not good enough?
- When to hand over to human? Or **fall back** to „safe/oblivious mode"?
- Can we learn from self-driving **cars**?



Conclusion

- Programmability and virtualization: algorithmic opportunities but also challenges
  - E.g.,: faster innovation, flexibilities in resource allocation, etc.
  - But, e.g.: performance isolation needs to be ensured across all involved resources, resulting resource allocation problems hard (open: good LP formulations, accounting for latencies, derandomization, special graphs, etc.)
- Security: more opportunities and challenges
  - Also faster innovation, but also new attack surface and potentially high impact
- AI opens interesting new opportunities
  - To deal with algorithmic complexities
  - To deal with modelling complexities
  - To find performance weaknesses
  - But also new challenges: how much control can we give away?

References

Predictable performance

On The Impact of the Network Hypervisor on Virtual Network Performance
Andreas Blenk, Arsany Basta, Wolfgang Kellerer, and Stefan Schmid.
IFIP Networking, Warsaw, Poland, May 2019.

Waypoint Routing in Special Networks
Saeed Akhoondian Amiri, Klaus-Tycho Foerster, Riko Jacob, Mahmoud Parham, and Stefan Schmid.
IFIP Networking, Zurich, Switzerland, May 2018.

Waypoint Routing

Walking Through Waypoints
Saeed Akhoondian Amiri, Klaus-Tycho Foerster, and Stefan Schmid.
13th Latin American Theoretical Informatics Symposium (**LATIN**), Buenos Aires, Argentina, April 2018.

Charting the Algorithmic Complexity of Waypoint Routing
Saeed Akhoondian Amiri, Klaus-Tycho Foerster, Riko Jacob, and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), 2018.

Virtual Network Embedding

Virtual Network Embedding Approximations: Leveraging Randomized Rounding
Matthias Rost and Stefan Schmid.
IEEE/ACM Transactions on Networking (**TON**), 2019.

Parametrized Complexity of Virtual Network Embeddings: Dynamic & Linear Programming Approximations
Matthias Rost, Elias Döhne, and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), January 2019.

Charting the Complexity Landscape of Virtual Network Embeddings (Best Paper Award)
Matthias Rost and Stefan Schmid.
IFIP Networking, Zurich, Switzerland, May 2018.

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

Security of Data Plane

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

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

AI Approaches

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

o'zapft is: Tap Your Network Algorithm's Big Data!
Andreas Blenk, Patrick Kalmbach, Stefan Schmid, and Wolfgang Kellerer.
ACM SIGCOMM 2017 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks (**Big-DAMA**), Los Angeles, California, USA, August 2017.

Tuple Space Explosion: A Denial-of-Service Attack Against a Software Packet Classifier
Levente Csikor, Dinil Mon Divakaran, Min Suk Kang, Attila Korosi, Balazs Sonkoly, David Haja, Dimitrios Pezaros, Stefan Schmid, and Gabor Retvari.
ACM **CoNEXT**, Orlando, Florida, USA, December 2019.