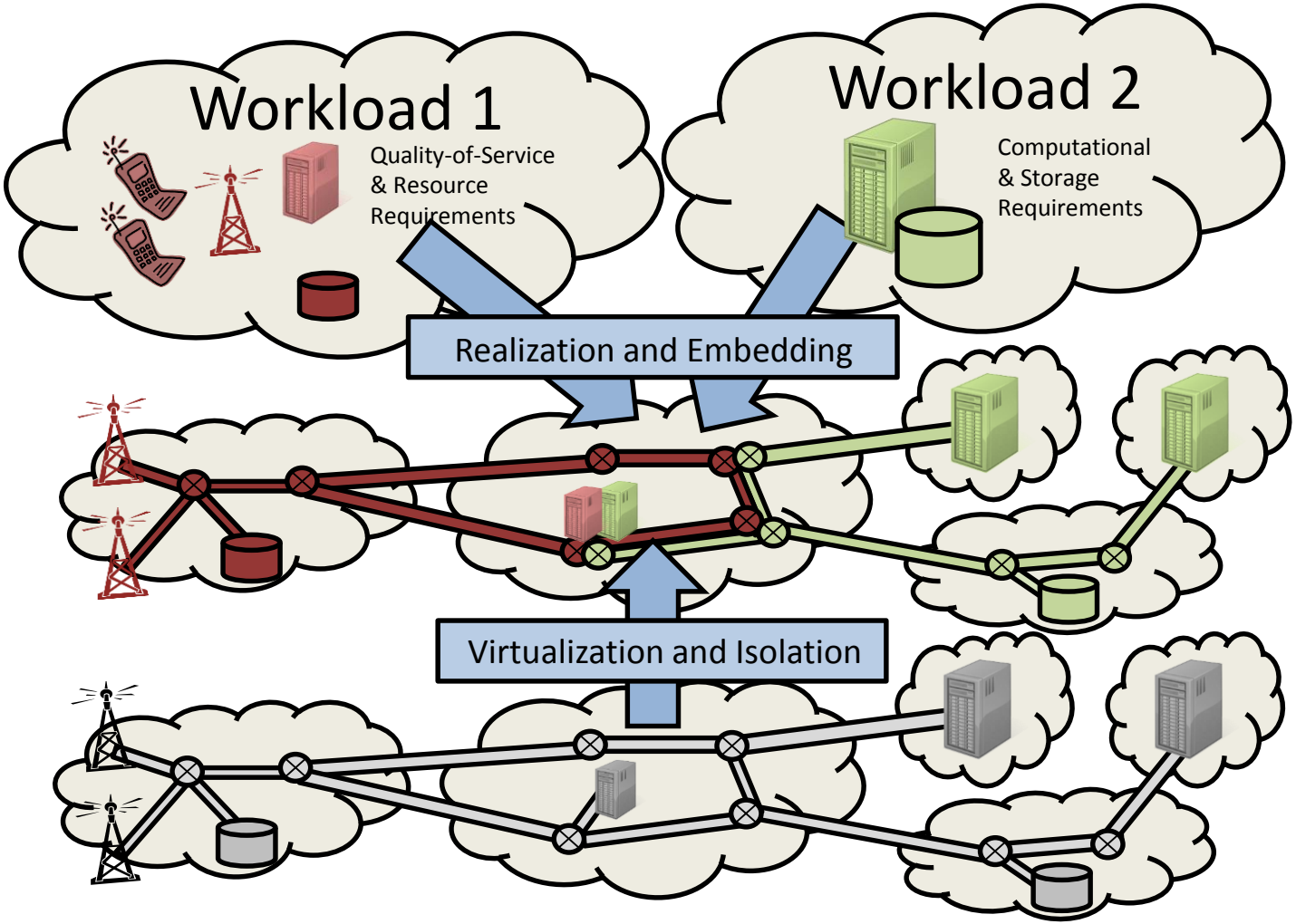
Network Slicing: Predictable Performance in Unpredictable Environment?

Stefan Schmid (University of Vienna, Austria)

The Promise: Network Slicing

- Flexible **resource allocation**: where and when most useful…
- … while providing **isolation**!
- Often: leveraging **virtualization**.


Workload 1
Quality-of-Service
& Resource
Requirements
Workload 2
Computational
& Storage
Requirements
Realization and Embedding
Virtualization and Isolation

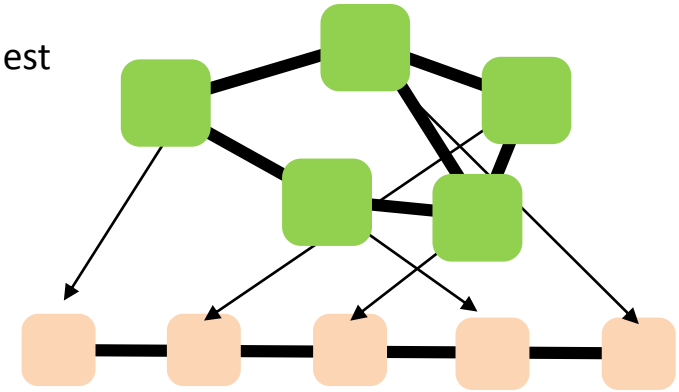
This Talk: 3 Challenges

- **Embedding** slices resource-efficiently is an open challenge
- But perhaps our **model is wrong** anyway? Practical challenges
- Performance isolation is one thing, **security** another

Challenge 1: Embedding

- Embedding problems are often **NP-hard**

Slice/VNet/Guest



Hard in many ways:

- Minimum Linear Arrangement (min sum embedding on a line)
- Subgraph isomorphism (cost=1 per virtual link: subgraph)
- Endpoints fixed: disjoint paths

- Possible solutions:
  - Exact exponential algorithms, e.g., formulate **Mixed Integer Program (MIP)**
  - Polynomial-time **approximation** algorithms, e.g., randomized rounding

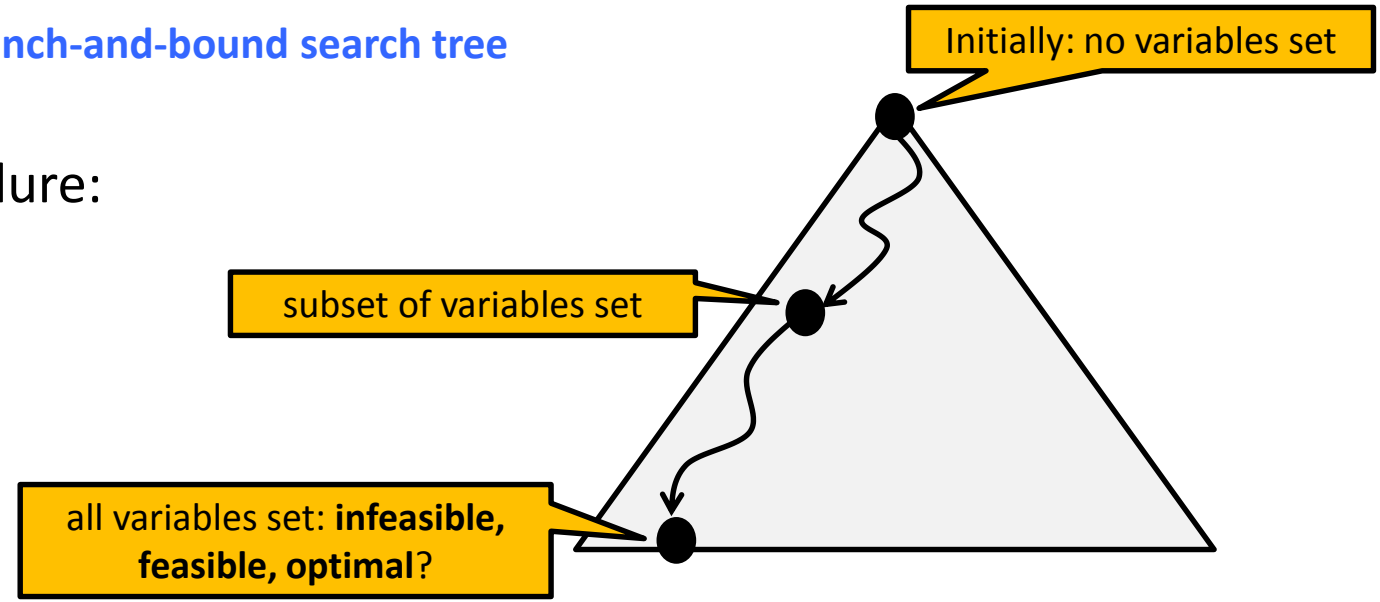
Formulating a Fast MIP

Formulating 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:

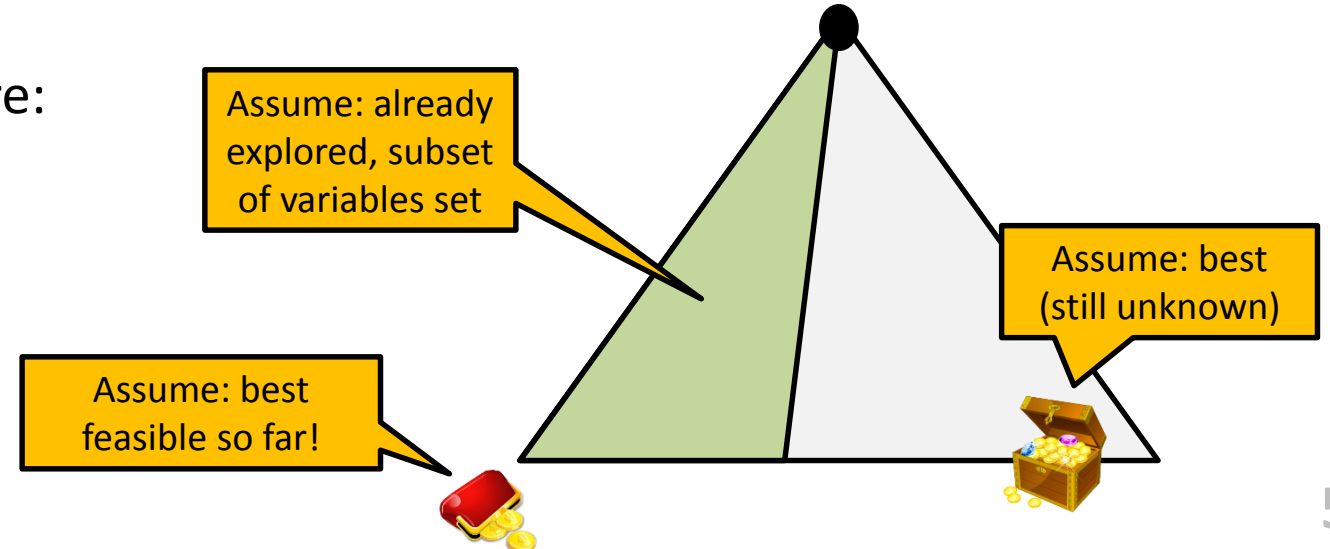


Formulating a Fast MIP

- 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:

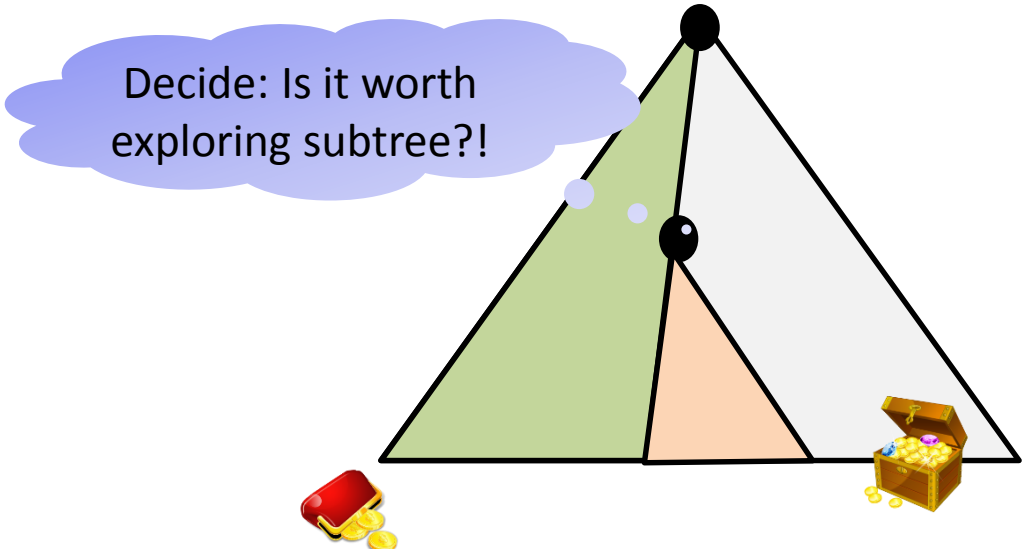


Formulating a Fast MIP

- 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:



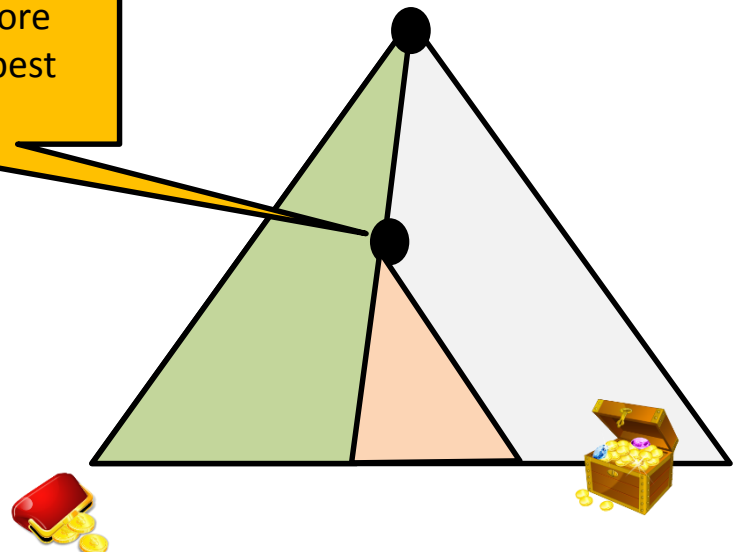
Formulating a Fast MIP

- 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 **bra**

Usual proced

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



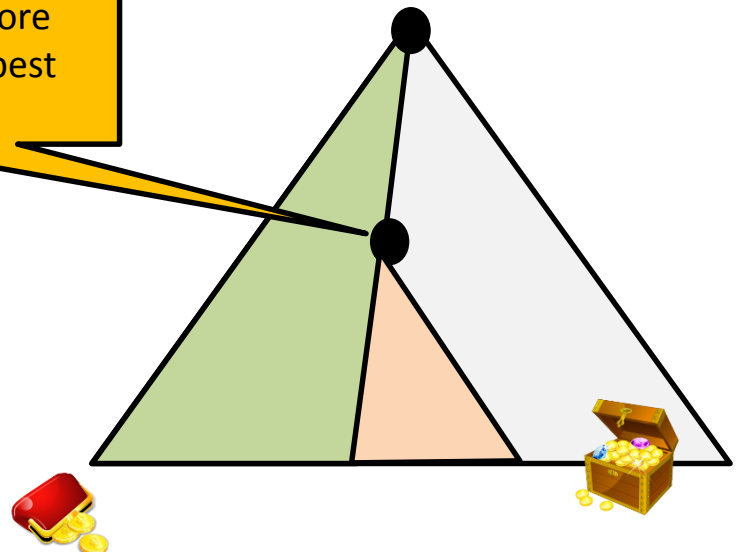
Formulating a Fast MIP

- 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 **bra**

Usual proced

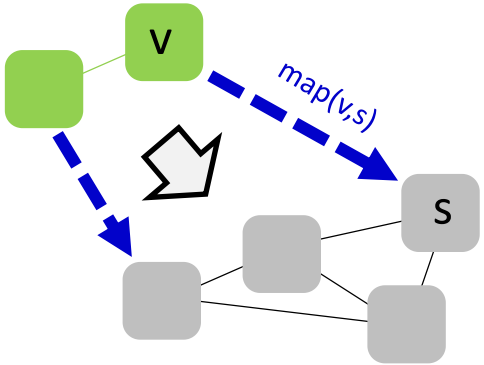
Usual trick: Relax! Solve LP (fast!), and if **relaxed solution** (more general!) **not better** then best solution so far: skip it!

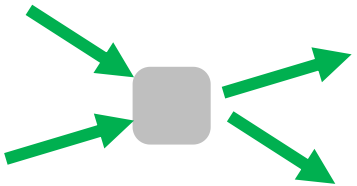


Bottomline: If MIP provides «good relaxations», large parts of the search space can be pruned.

MIP: A Formulation

- „Usual MIP“
  - Binary variables **map(v,s)** to map virtual node v to substrate node s
  - Introduce **flow variables** for paths
  - Ensure **flow conservation**: all flow entering a node must leave the node, unless source or destination

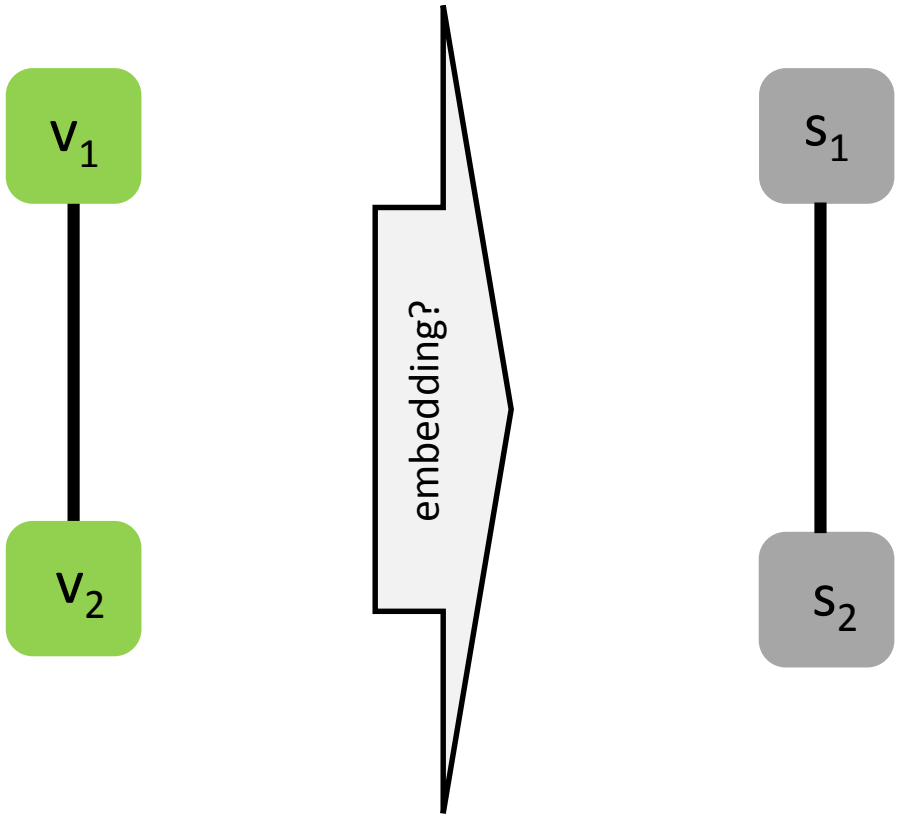




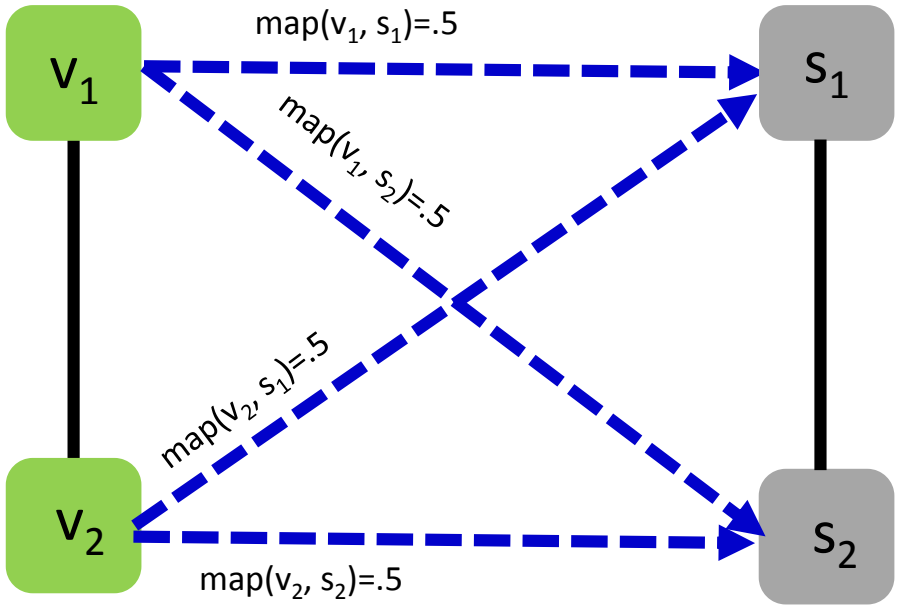
$\Sigma_{u->v} f_{uv} = \Sigma_{v->w} f_{vw}$

In Out

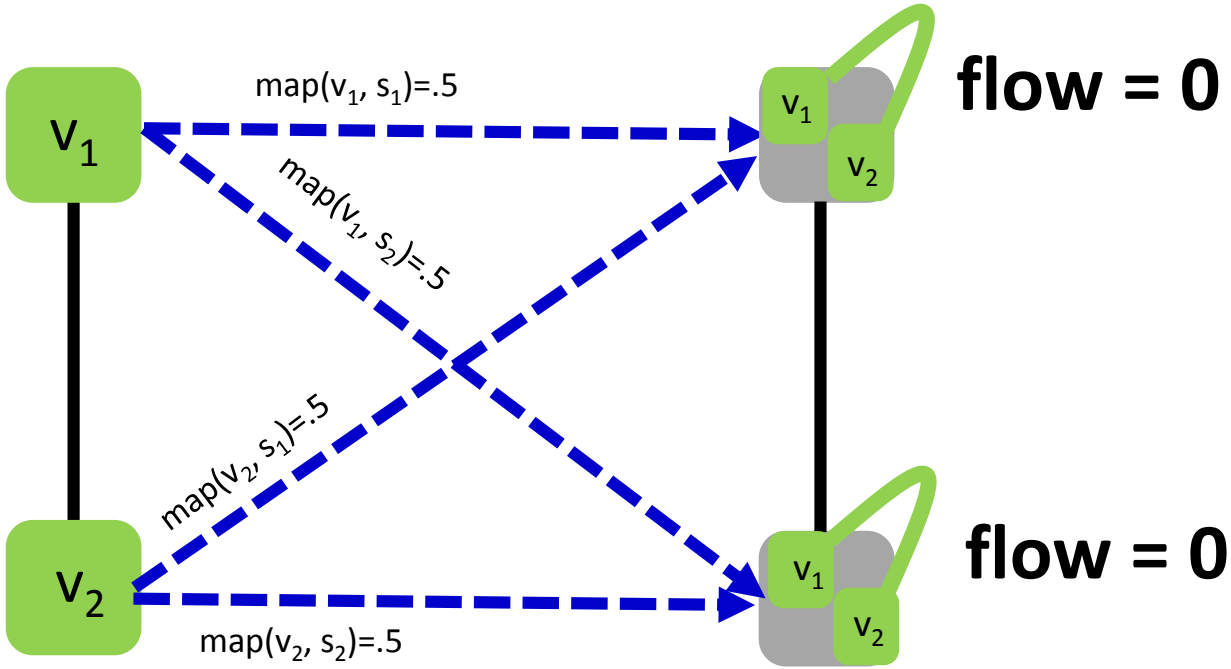
What will happen in this case?



What will happen in this case?



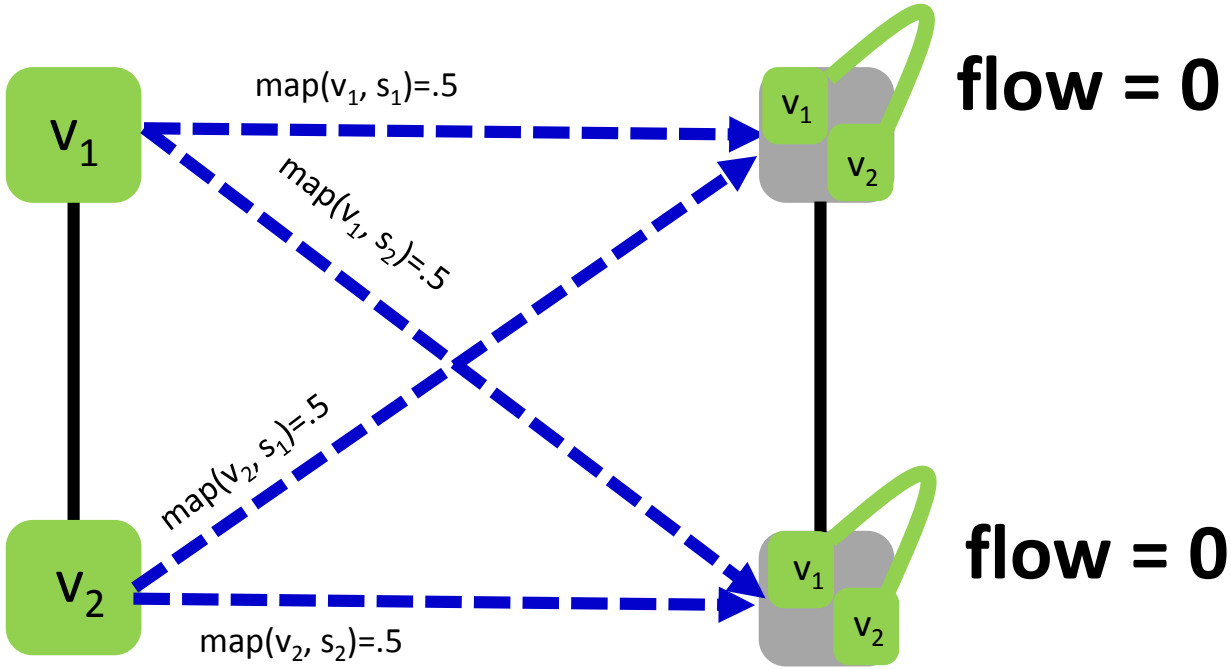
What will happen in this case?



Minimal flow = 0: fulfills flow conservation but relaxation useless!
Does not provide any lower bound or indication of good mapping!

What will happen in this case?

The MIP formulation matters!



Minimal flow = 0: fulfills flow conservation but relaxation useless!
Does not provide any lower bound or indication of good mapping!

Another Approach: Approximation

- MIPs take **super-polynomial** time in worst case
- Alternative: polynomial-time **approximation**
- E.g., **randomized rounding:**
  - Formulate MIP resp. ILP
  - Compute relaxation: relaxed solutions are linear combinations of elementary solutions
  - Probabilistically choose any of the elementary solutions based on their weights

Idea: Approx Using MCF Formulation

For example, VNEP based on standard **Multi-Commodity Flow (MCF)** formulation

Formulation 1: Classic MCF Formulation for the VNEP

$$\max \sum_{r \in \mathcal{R}} b_r x_r \tag{1}$$

$$\sum_{u \in V_S^{r,i}} y_{r,i}^u = x_r \qquad \forall r \in \mathcal{R}, i \in V_r \tag{2}$$

$$\sum_{u \in V_S \setminus V_S^{r,i}} y_{r,i}^u = 0 \qquad \forall r \in \mathcal{R}, i \in V_r \tag{3}$$

$$\begin{bmatrix} \sum_{(u,v) \in \delta^+(u)} z_{r,i,j}^{u,v} \\ -\sum_{(v,u) \in \delta^-(u)} z_{r,i,j}^{v,u} \end{bmatrix} = \begin{bmatrix} y_{r,i}^u \\ -y_{r,j}^u \end{bmatrix} \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ u \in V_S \end{bmatrix} \tag{4}$$

$$z_{r,i,j}^{u,v} = 0 \qquad \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ (u,v) \in E_S \setminus E_S^{r,i,j} \end{bmatrix} \tag{5}$$

$$\sum_{i \in V_r, \tau_r(i) = \tau} d_r(i) \cdot y_{r,i}^u = a_r^{\tau,u} \qquad \forall r \in \mathcal{R}, (\tau, u) \in R_S^V \tag{6}$$

$$\sum_{(i,j) \in E_r} d_r(i,j) \cdot z_{r,i,j}^{u,v} = a_r^{u,v} \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{7}$$

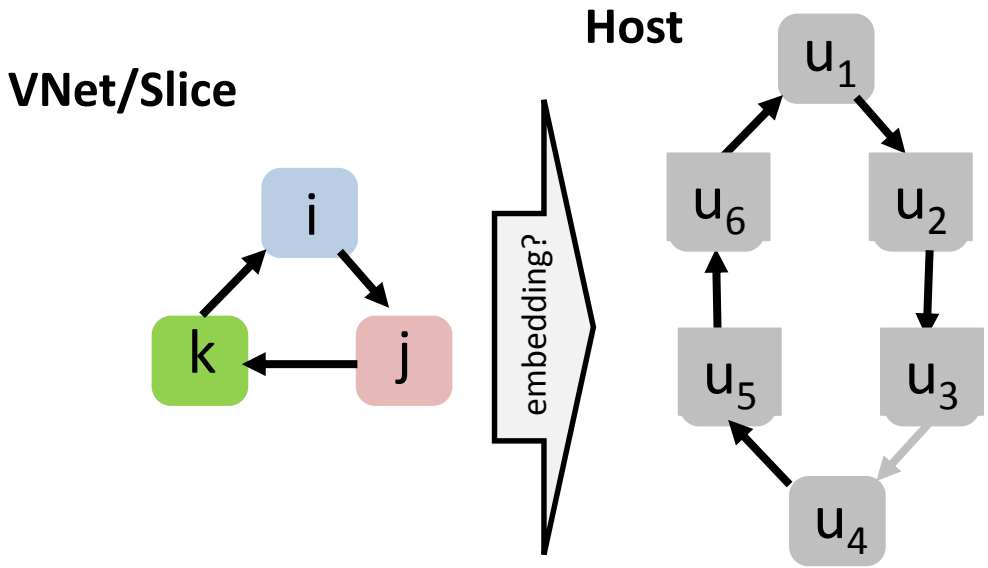
$$\sum_{r \in \mathcal{R}} a_r^{x,y} \leq d_S(x,y) \; \forall (x,y) \in R_S \tag{8}$$

Randomized Rounding Can Fail

- **Good news:** works on line and tree requests
  - E.g., approximate service chain embeddings
  - Apply Raghavan and Thompson

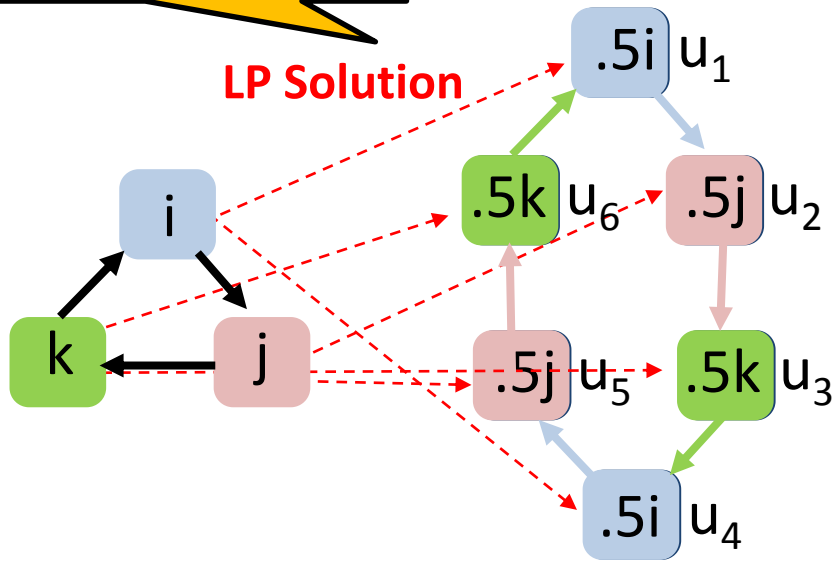
- **Bad news:** for requests which are not acyclic, the integrality gap can be infinite and the problem not decomposable
  - LP solutions to classic MCF formulation can no longer be decomposed into convex combinations of valid mappings

Randomized Rounding Can Fail



Randomized Rounding Can Fail

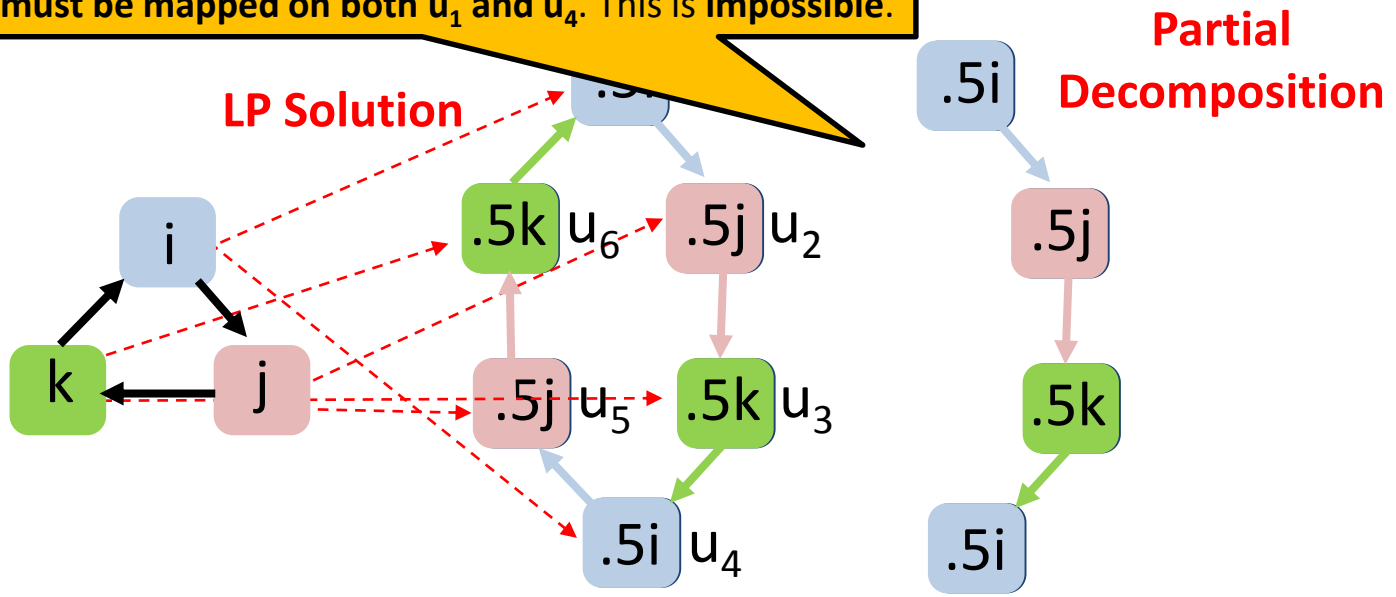
Valid LP solution: virtual node mappings sum to 1 and each virtual node connects to its neighboring node **with half a unit of flow**…



Relaxations of classic MCF formulation **cannot be decomposed** into convex combinations of valid mappings (so we **need different formulations**!)

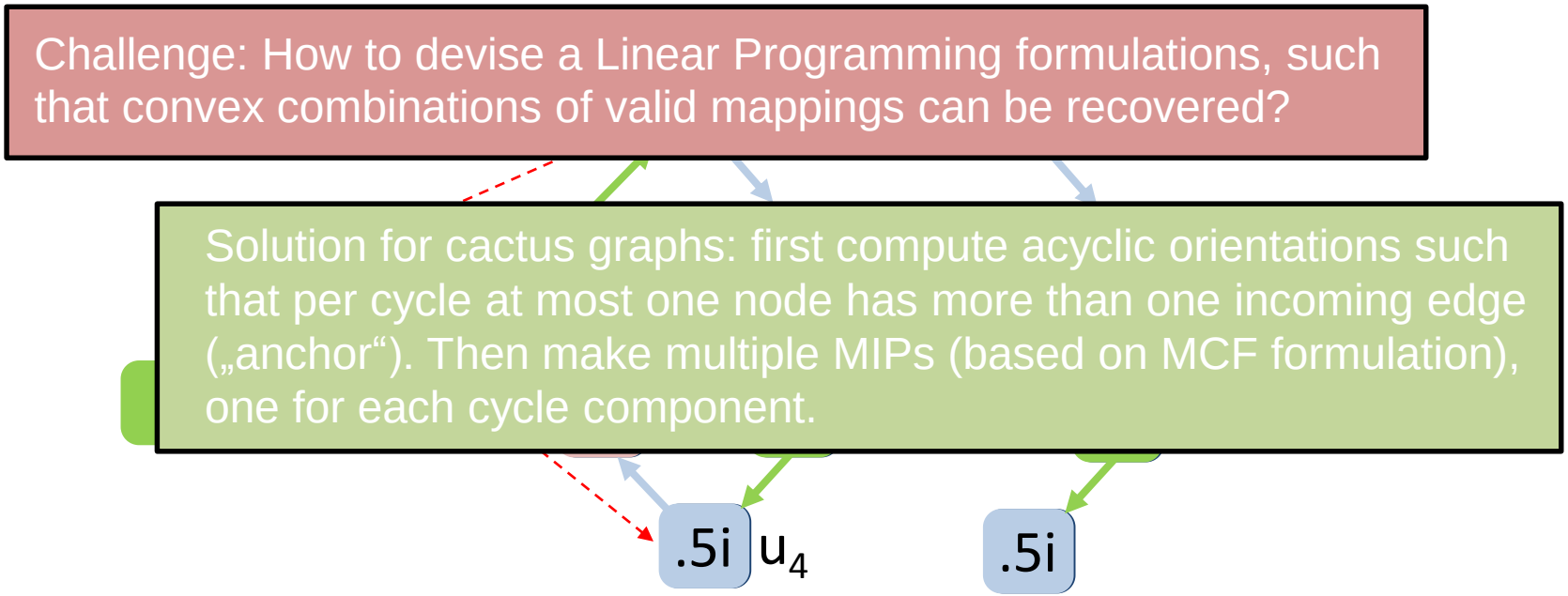
g Can Fail

Impossible to decompose and extract **any single valid mapping**.
Intuition: Node i is mapped to u_1 and **the only neighboring node** that hosts j is u_2, so i must be fully mapped on u_1 and j on u_2. Similarly, k must be mapped on u_3. But flow of virtual edge (k,i) leaving u_3 only leads to u_4, so **i must be mapped on both u_1 and u_4**. This is **impossible**.



Relaxations of classic MCF formulation **cannot be decomposed** into convex combinations of valid mappings (so we **need different formulations**!)

Randomized Rounding Can Fail

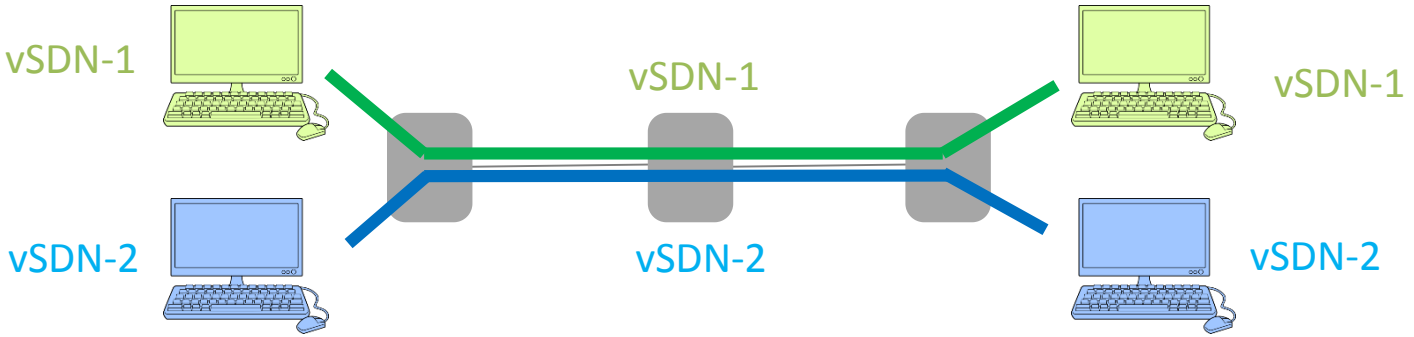


Relaxations of classic MCF formulation **cannot be decomposed** into convex combinations of valid mappings (so we **need different formulations**!)

Challenge 2: Model

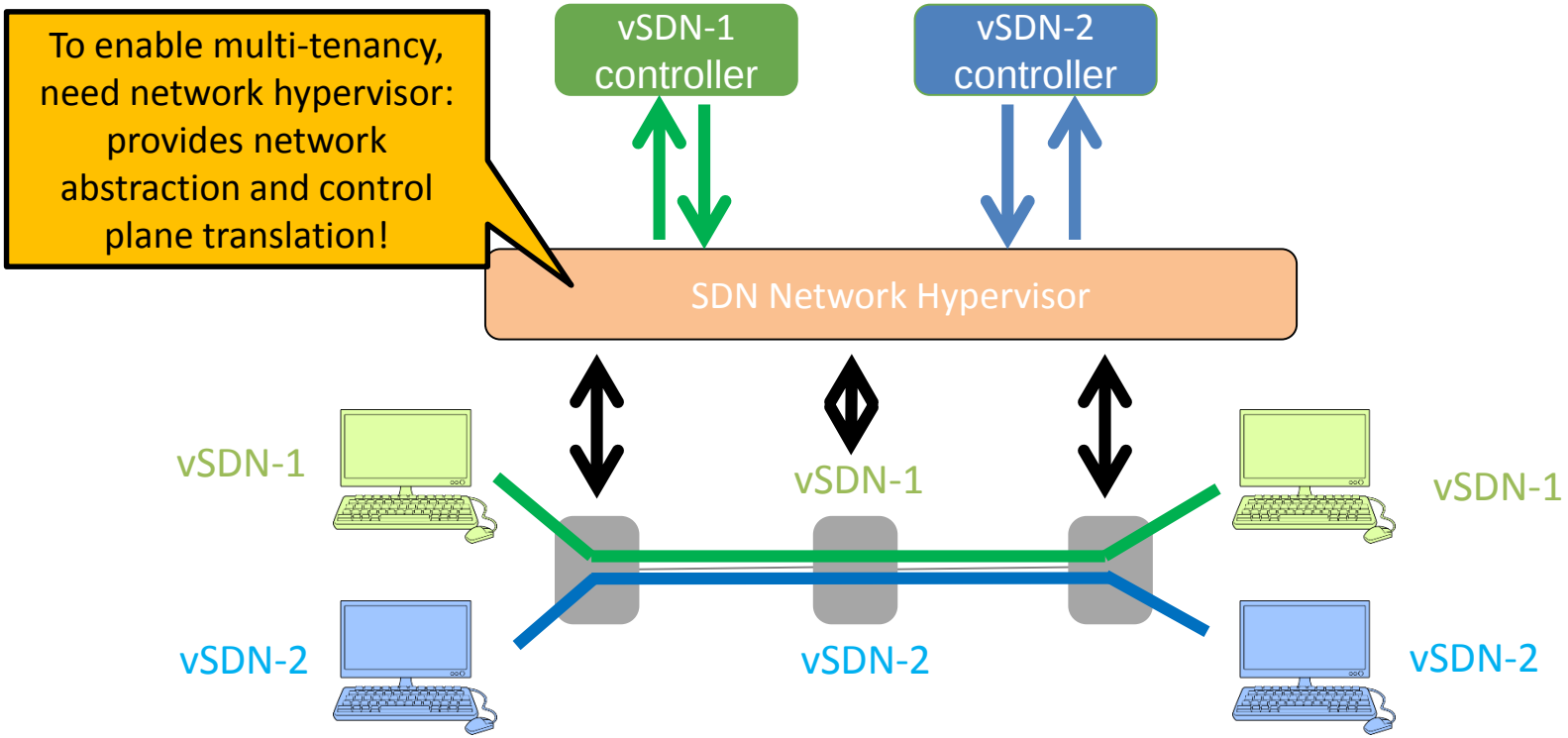
How good are your models anyway?!

- **Predictable performance** is about more than just bandwidth reservation



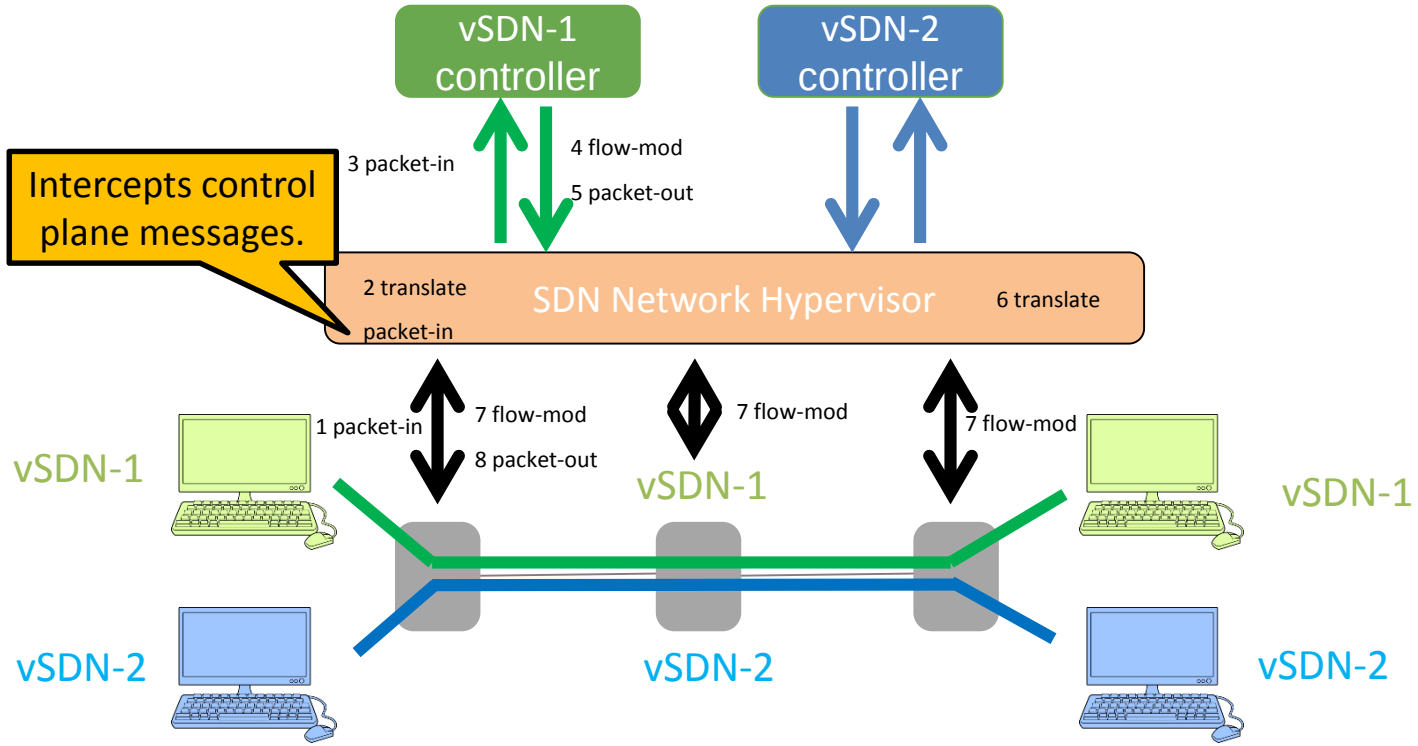
An Experiment: 2 vSDNs with bw guarantee!

Models Must Be More Complex



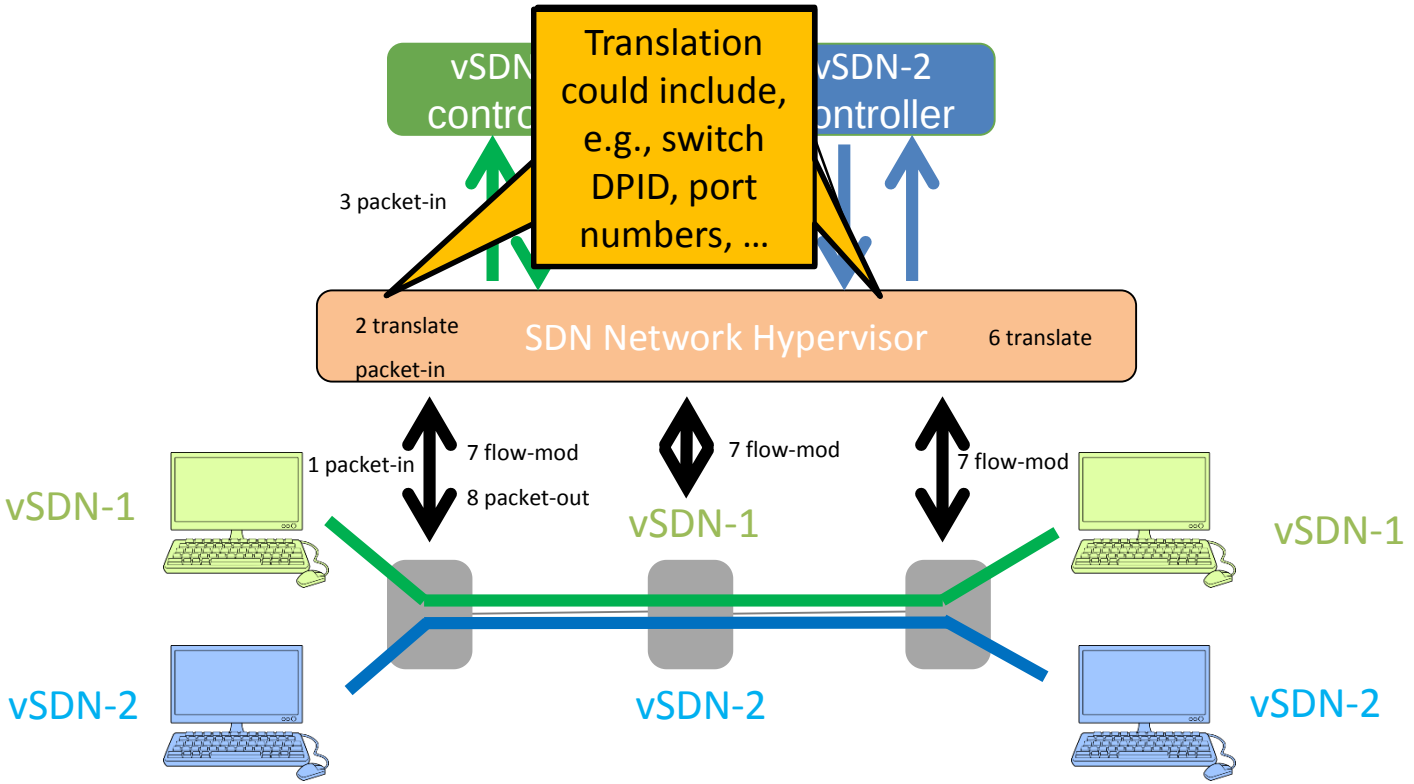
An Experiment: 2 vSDNs with bw guarantee!

Models Must Be More Complex



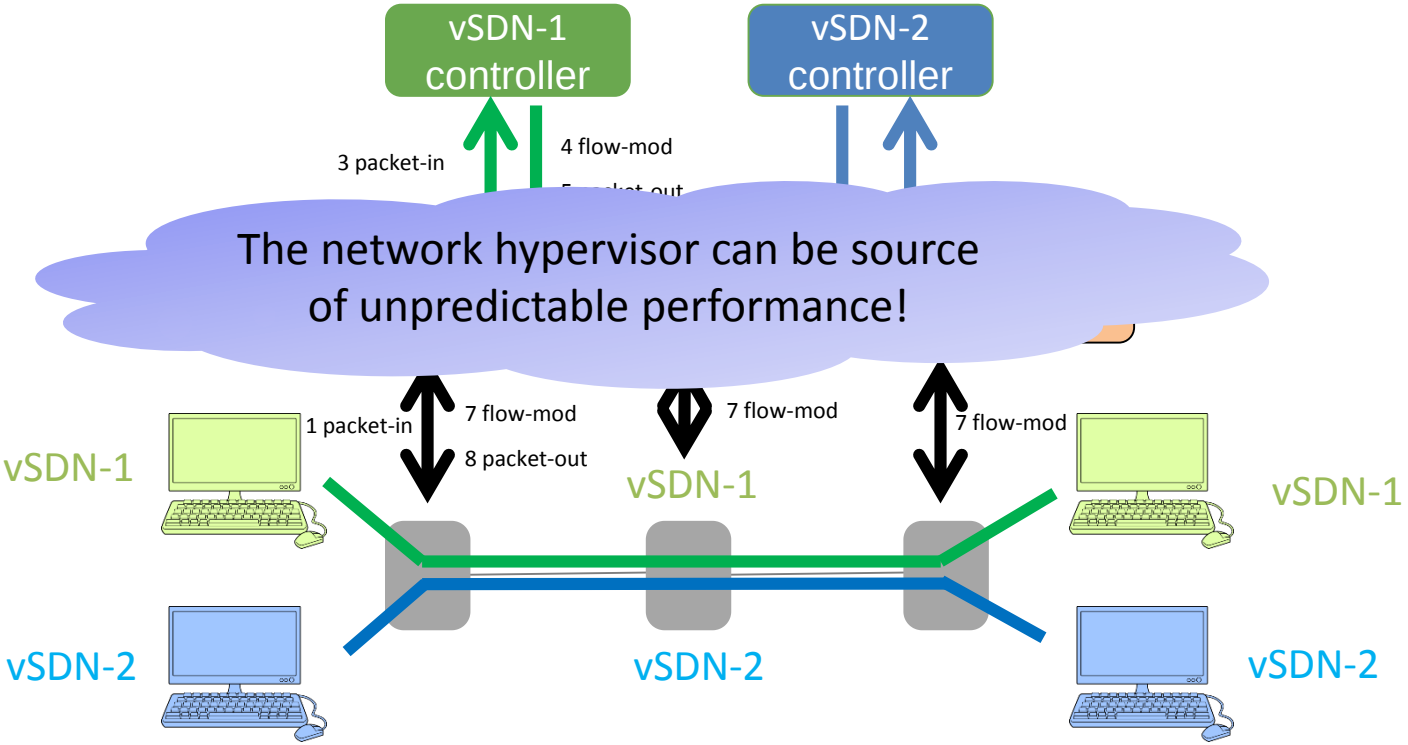
An Experiment: 2 vSDNs with bw guarantee!

Models Must Be More Complex



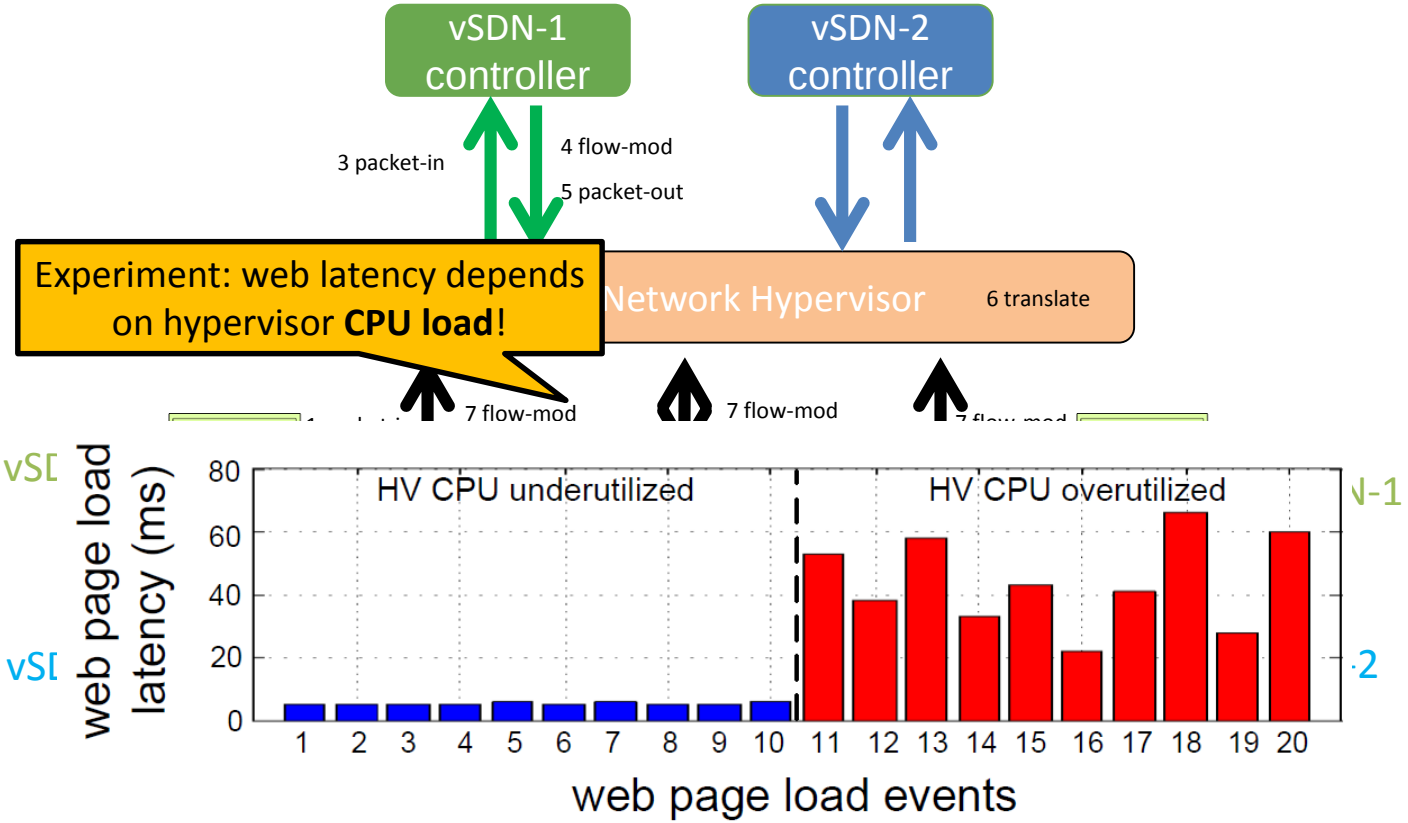
An Experiment: 2 vSDNs with bw guarantee!

Models Must Be More Complex

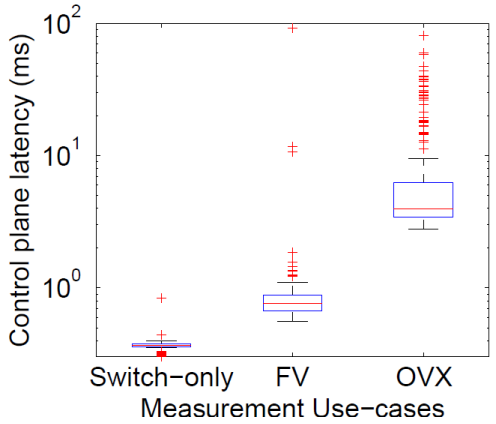


An Experiment: 2 vSDNs with bw guarantee!

Models Must Be More Complex



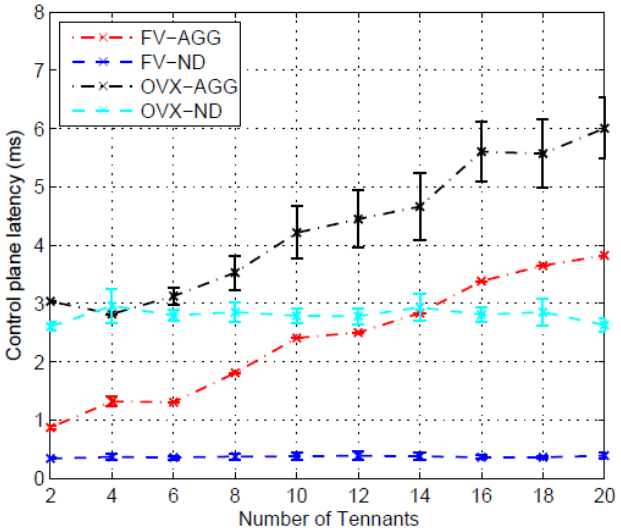
Need to Know Your Network Hypervisor



Performance also depends on hypervisor type…

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

… number of tenants…

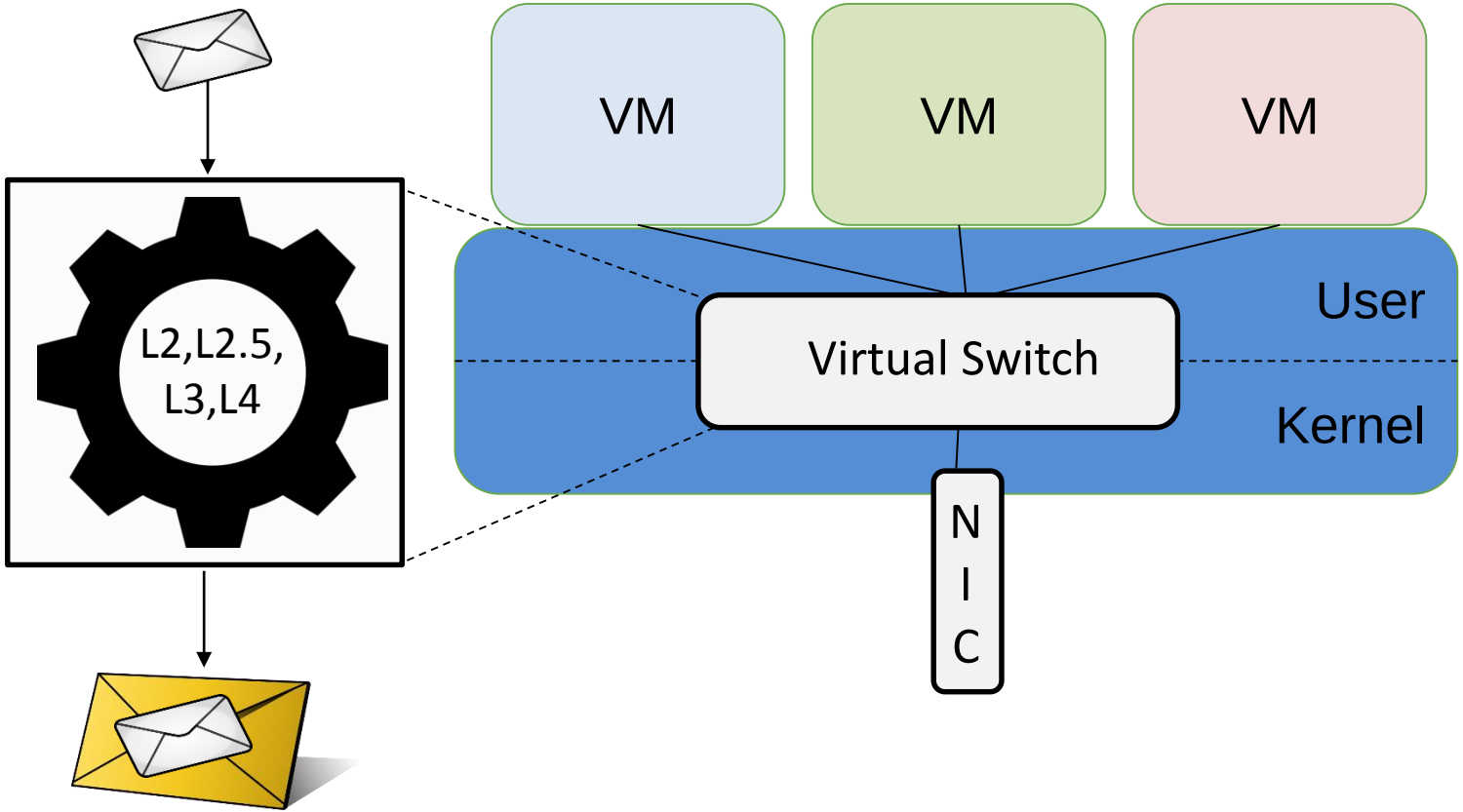


Challenge 3: Security

- **Performance isolation** between slices is essential for providing a predictable performance
- Can be achieved using **virtualization**
- However, **isolation** between slices is also crucial for security

Virtual Switches are Complex, e.g.: (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



A Threat: Packet Parser

- More and more **complex** (unified parsing for speed)
- **Faces the attacker**: first component to receive adversarial inputs
- Virtual switches run with high security **privileges**
- Case study:
  - Fuzzing 2% of OVS code
  - Bugs e.g. in MPLS

Discussion

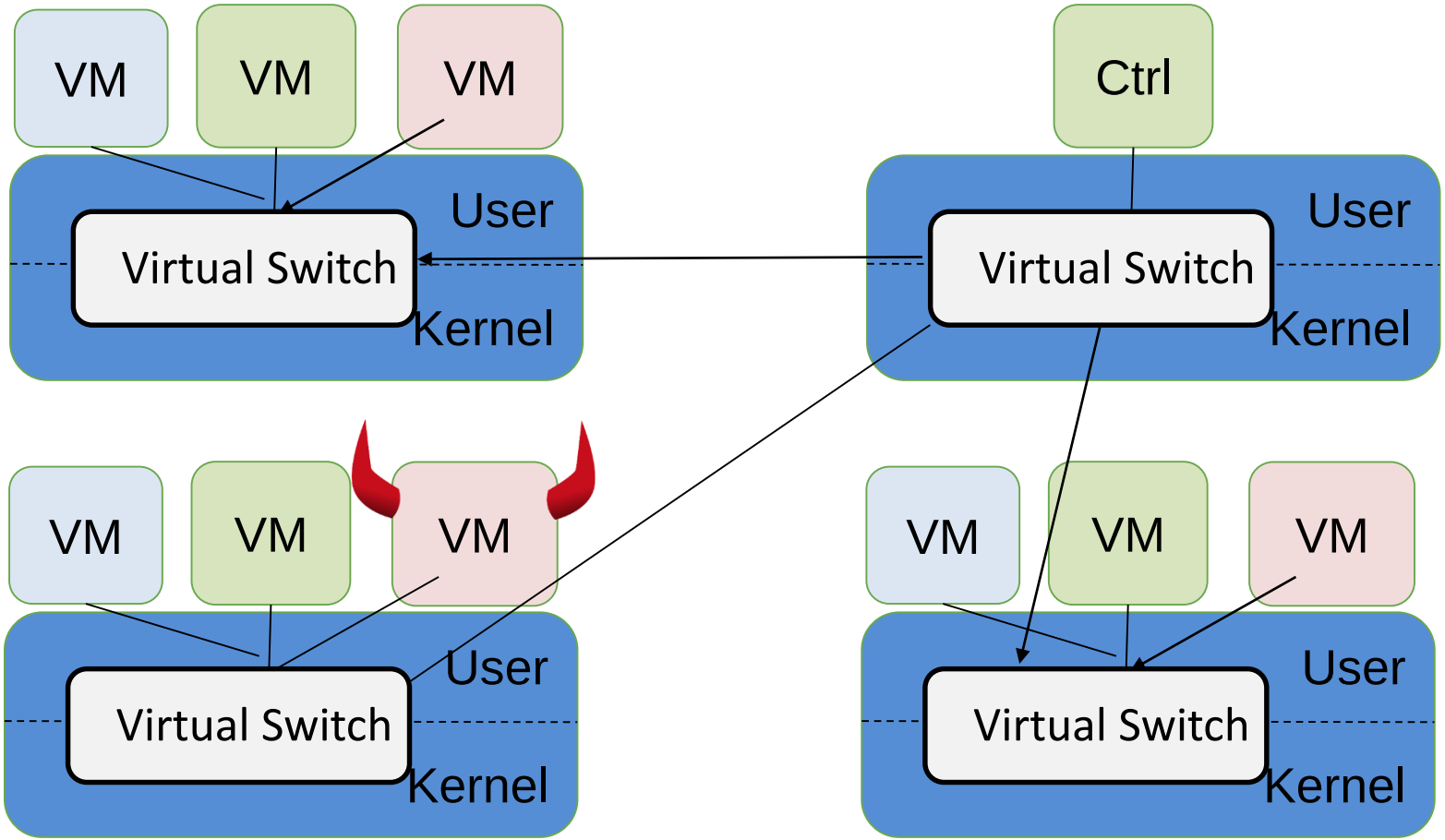
- **Issue 1:** Increases **attack surface** and moves it **closer** to adversary

- **Issue 2:** **Cheap** to exploit
  - Use some standard fuzzer to find bugs
  - Rent a VM in the cloud (low cost!)

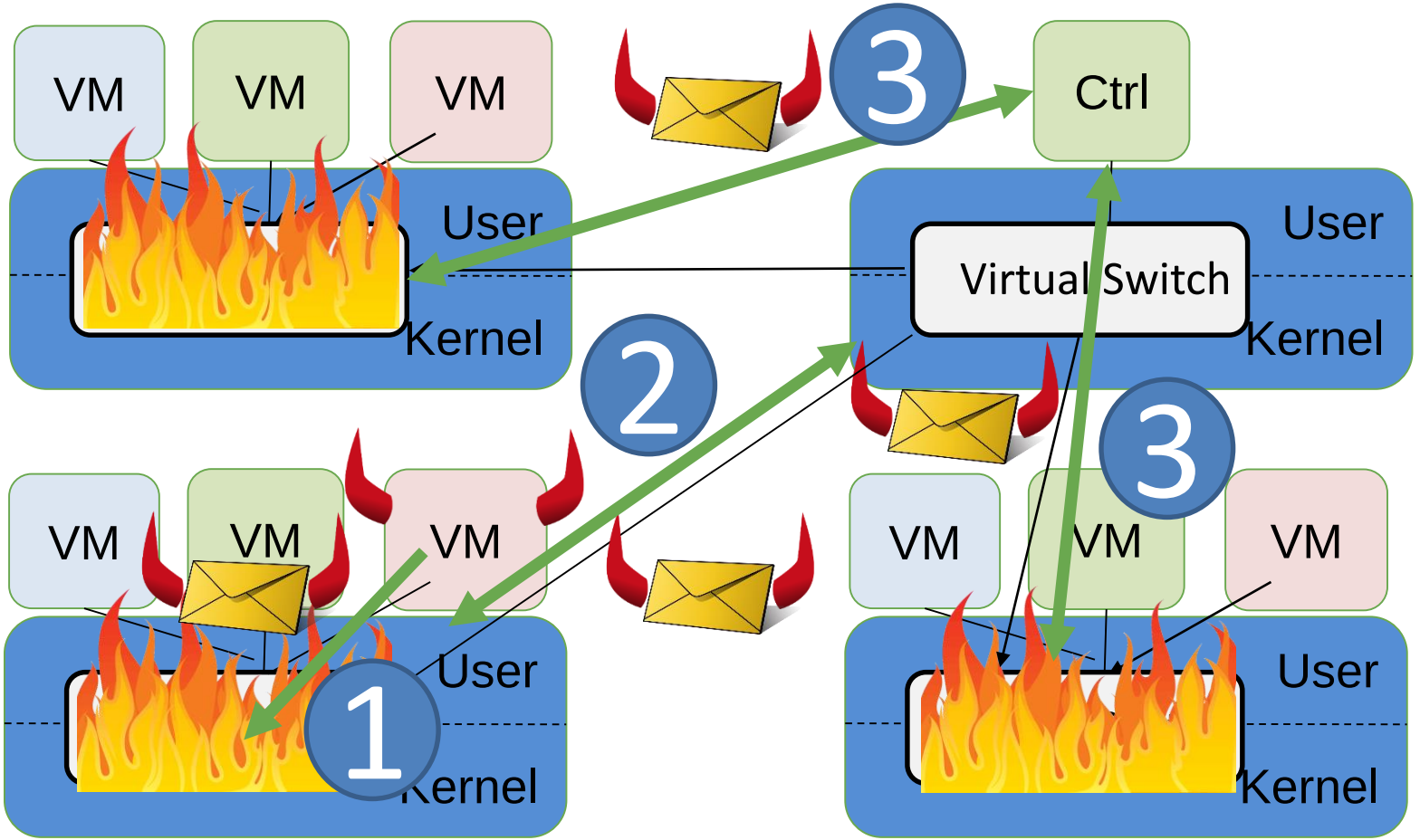
- **Issue 3:** Huge **impact**
  - Collocation: Do to virtualization, can attack collocated applications
  - Logical Centralization: Can spread a **worm**, e.g., over logically centralized controller

New threat model: The **vAMP Attack**

Compromising the Cloud



Compromising the Cloud



Conclusion

- Challenge 1: Fast algorithms for slice resource allocation
- Challenge 2: Good models
- Challenge 3: Security

Further Reading

- Hardness of embedding:

  Charting the Complexity Landscape of Virtual Network Embeddings
  Matthias Rost and Stefan Schmid. **IFIP Networking**, Zurich, Switzerland, May 2018.

- Randomized rounding and decomposability:

  Virtual Network Embedding Approximations: Leveraging Randomized Rounding
  Matthias Rost and Stefan Schmid. **IFIP Networking**, Zurich, Switzerland, May 2018.

- Modeling and hypervisor interference:

  Logically Isolated, Actually Unpredictable? Measuring Hypervisor Performance in Multi-Tenant SDNs
  Arsany Basta, Andreas Blenk, Wolfgang Kellerer, and Stefan Schmid. ArXiv Technical Report, May 2017.

- Isolation and security:

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

  The vAMP Attack: Taking Control of Cloud Systems via the Unified Packet Parser
  Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid. 9th ACM Cloud Computing Security Workshop (**CCSW**), collocated with ACM CCS, Dallas, Texas, USA, November 2017.