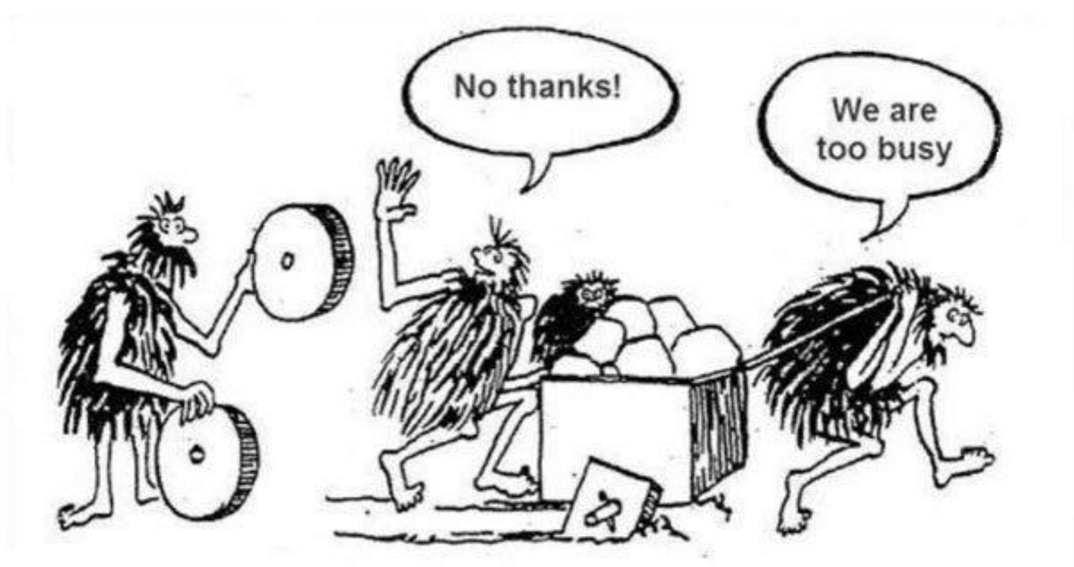
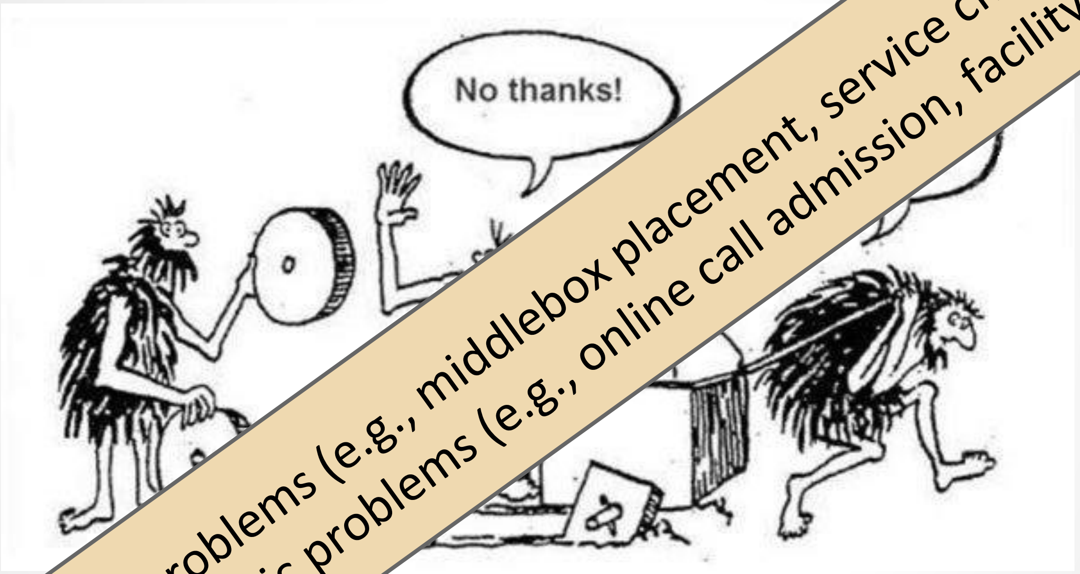
Algorithmic Challenges in Network Function Virtualized Networks

Stefan Schmid

TU Berlin & Telekom Innovation Labs (T-Labs)

Joint work mainly with
Tamás Lukovszki, Matthias Rost, Carlo Fürst

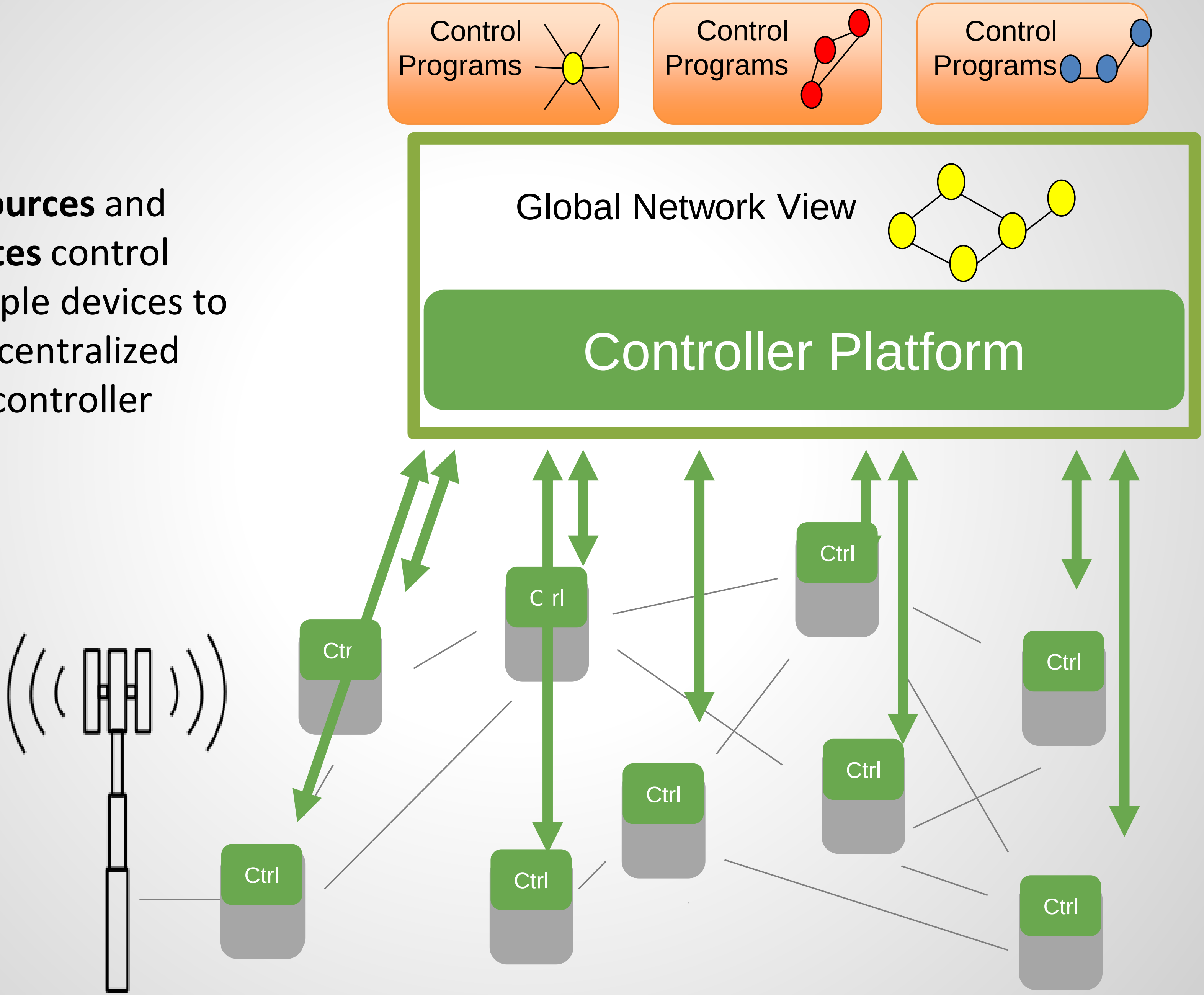

No thanks!
We are too busy



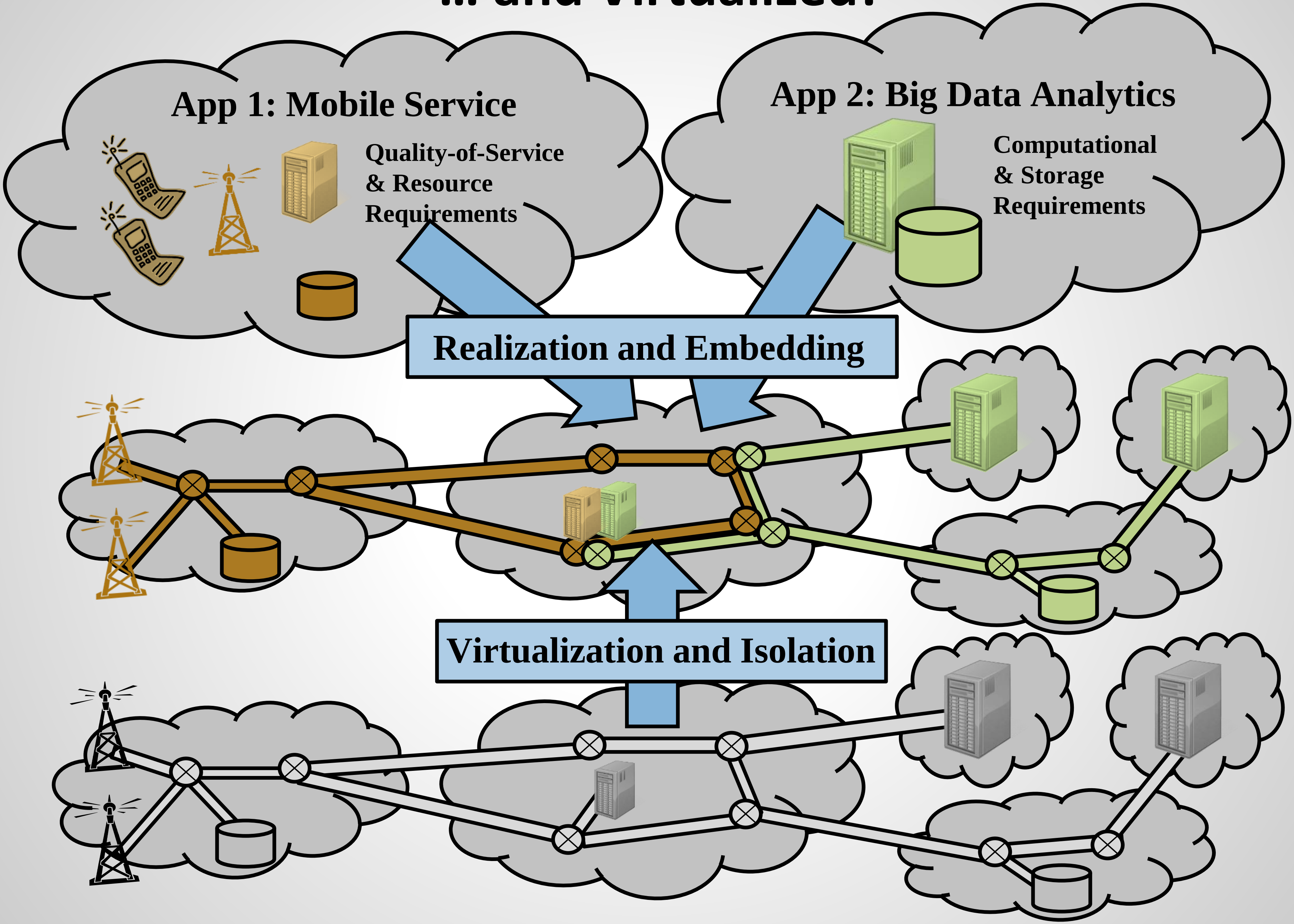
Algorithmic NFV problems (e.g., middlebox placement, service chain embedding) are related to classic problems (e.g., online call admission, facility location)!

Flexible Networked Systems: Programmable...

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



… and Virtualized!



App 1: Mobile Service

Quality-of-Service
& Resource
Requirements

App 2: Big Data Analytics

Computational
& Storage
Requirements

Realization and Embedding

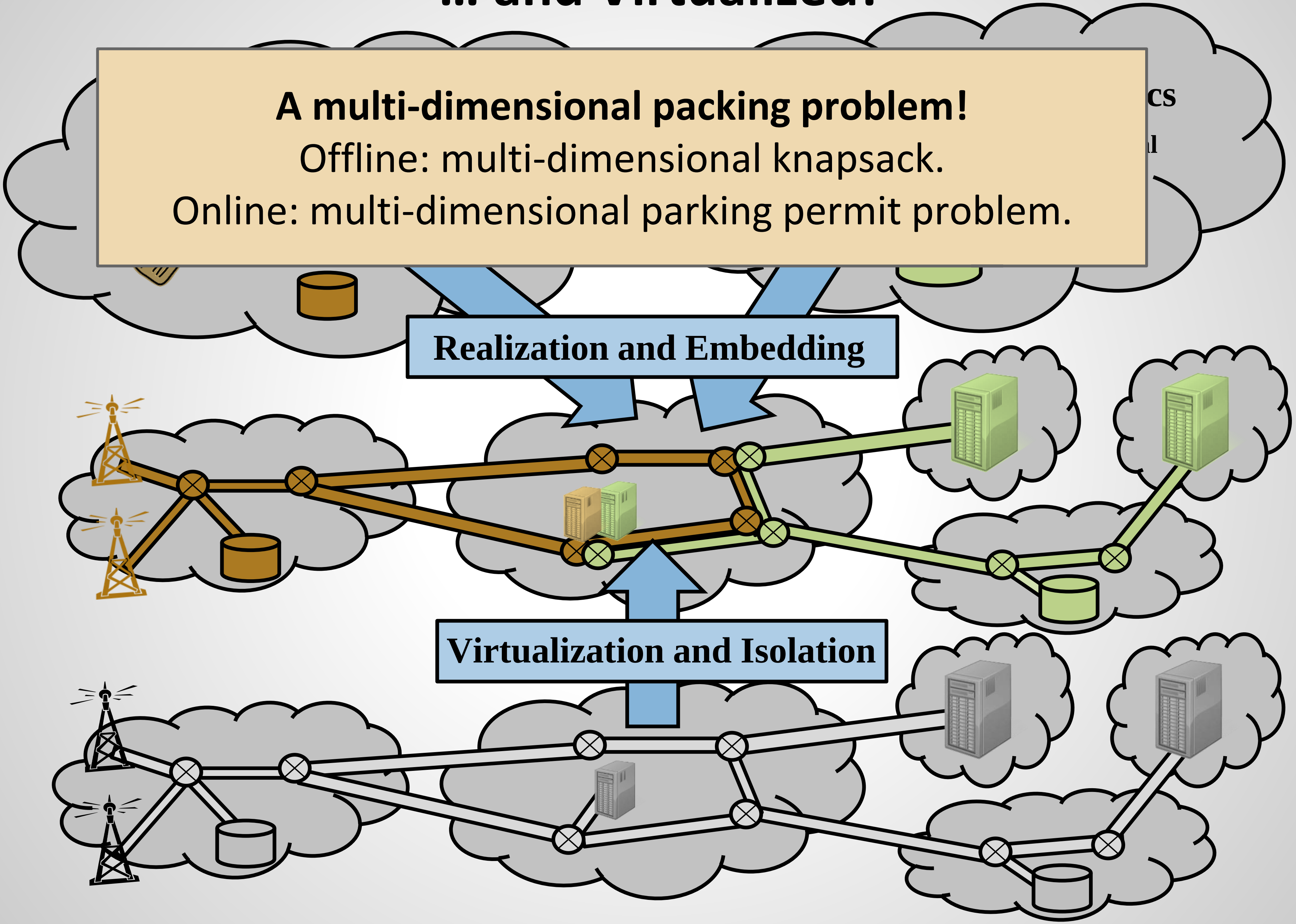
Virtualization and Isolation

… and Virtualized!

A multi-dimensional packing problem!

Offline: multi-dimensional knapsack.

Online: multi-dimensional parking permit problem.



Realization and Embedding

Virtualization and Isolation

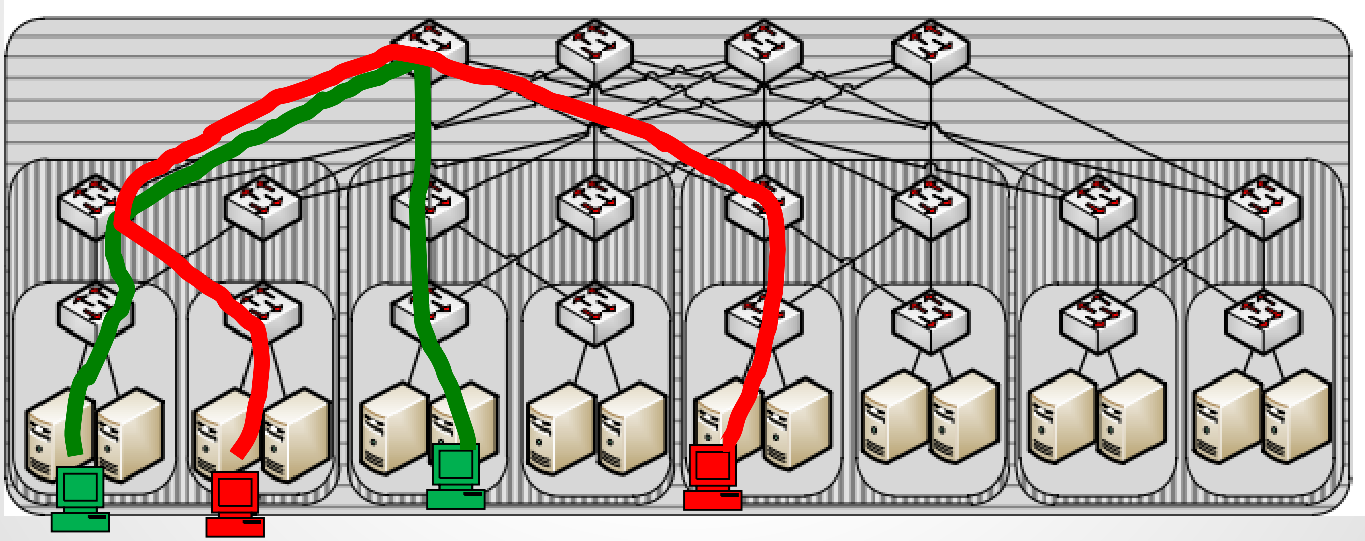
It’s a Great Time to Be a Scientist

”We are at an interesting inflection point!”

Keynote by George Varghese at SIGCOMM 2014

How to Exploit Flexibilities?
Example 1: Virtual Network Embedding

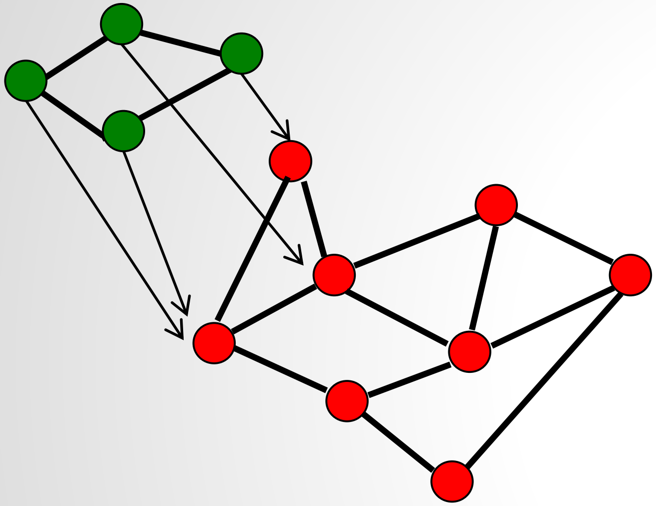
- Flexible embedding of virtual machines…
- … and their interconnecting network.



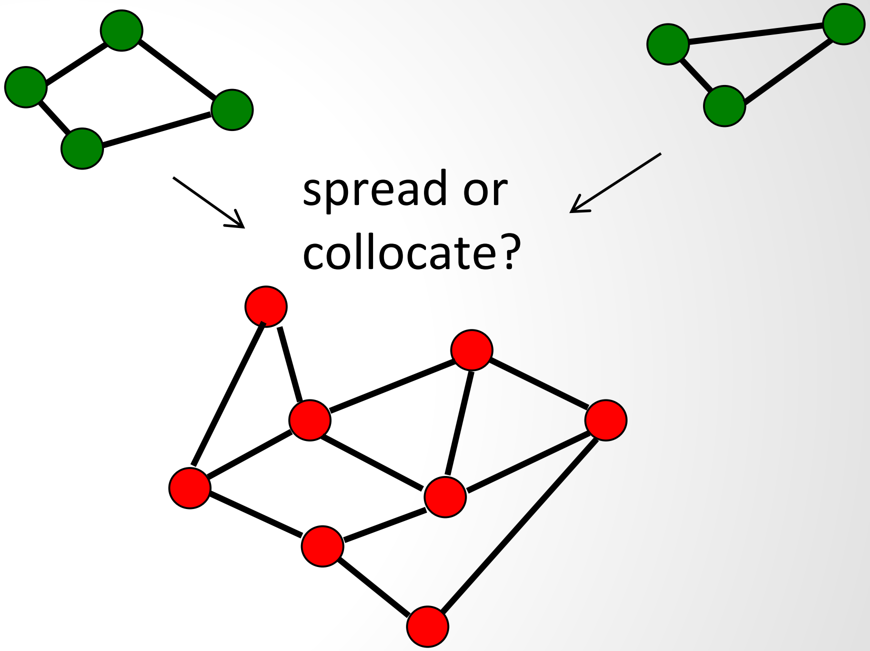
- How to max utilization? A **network embeddig problem**!

Flavors of VNet Embedding Problems (VNEP)

Minimize embedding footprint of a single VNet :

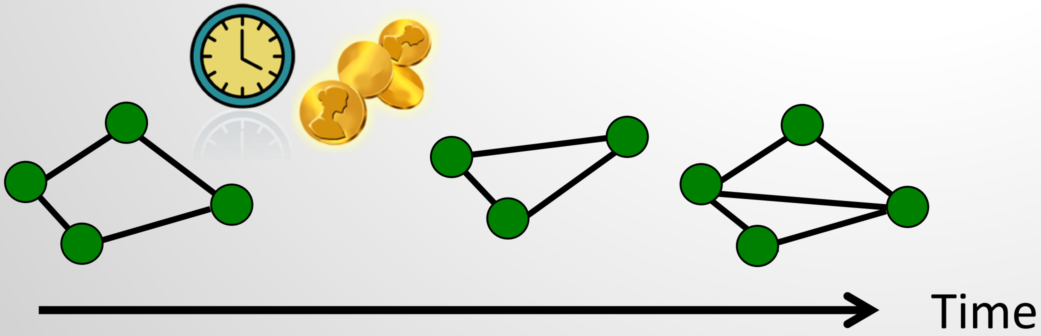


Minimize max load of multiple VNets or collocate to save energy:



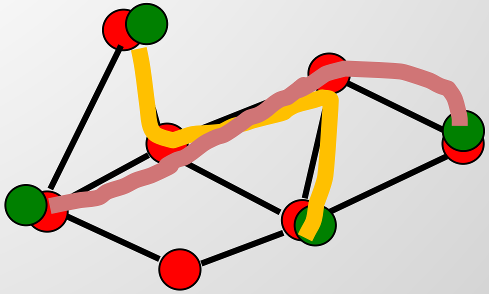
spread or collocate?

Maximize profit over time:



Time

Endpoints fixed:



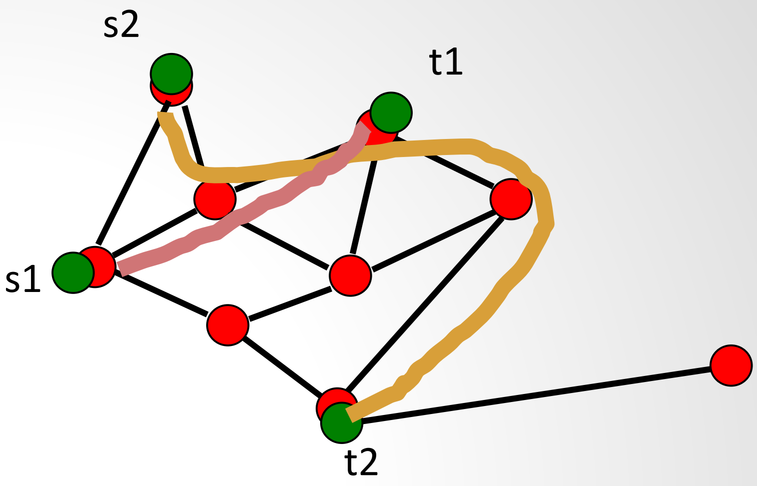
A ticket at a cloud hosting company…

«A tenant requested an upgrade,
needs 30 more VMs.
Why did the request fail?
There are hundreds of idle cores!»

Let’s Exploit Allocation Flexibilities to Maximize Utilization

Let’s Exploit Allocation Flexibilities to Maximize Utilization

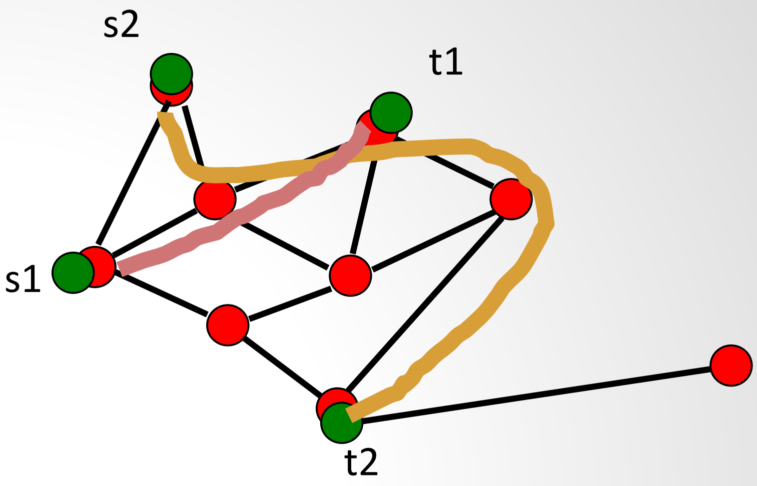
Start simple: exploit flexible routing between given VMs



Let’s Exploit Allocation Flexibilities to Maximize Utilization

Start simple: exploit flexible routing between given VMs

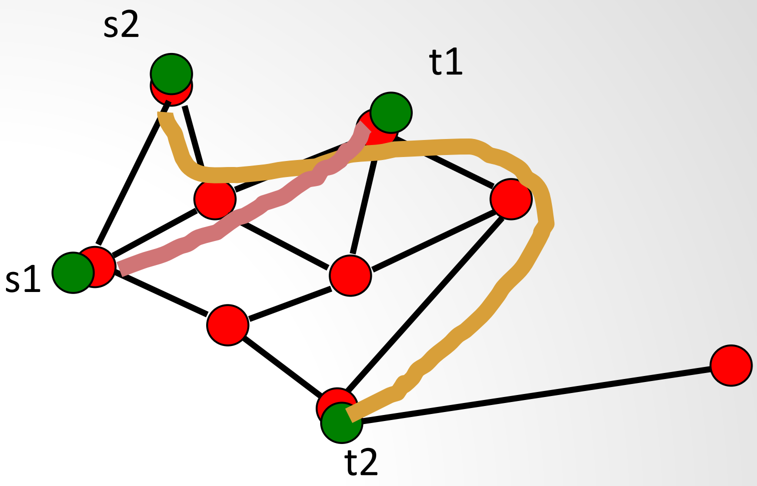
- ❑ Integer multi-commodity flow problem with 2 flows?



Let’s Exploit Allocation Flexibilities to Maximize Utilization

Start simple: exploit flexible routing between given VMs

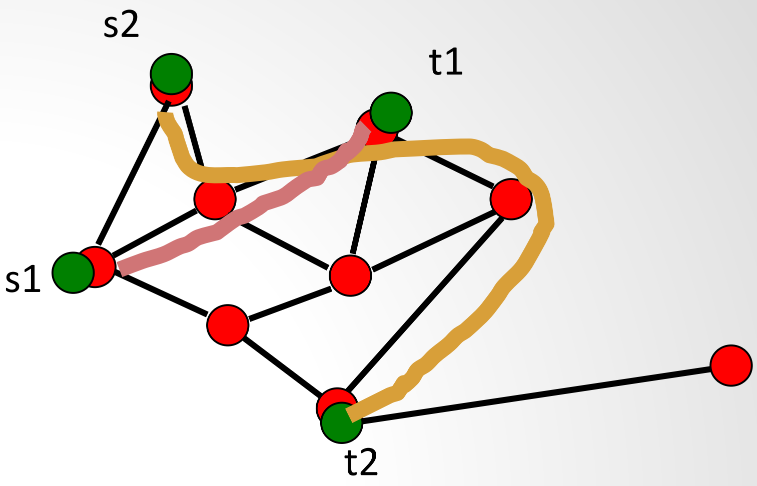
- ❑ Integer multi-commodity flow problem with 2 flows?
- ❑ Oops: NP-hard

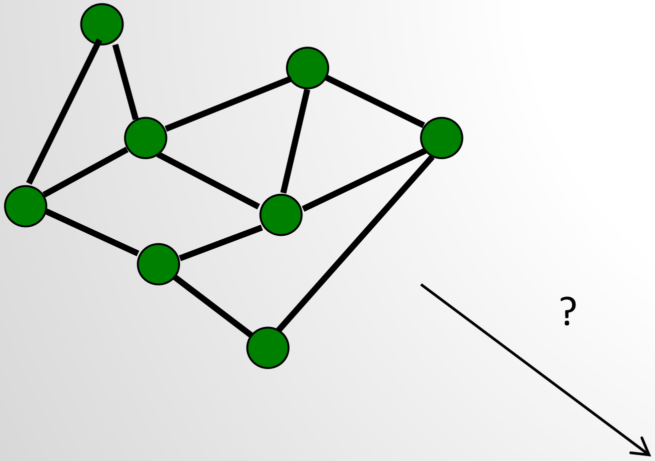


Let’s Exploit Allocation Flexibilities to Maximize Utilization

Start simple: exploit flexible routing between given VMs

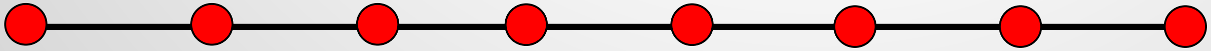
- Integer multi-commodity flow problem with 2 flows?
- Oops: NP-hard





Forget about paths: exploit VM placement flexibilities!

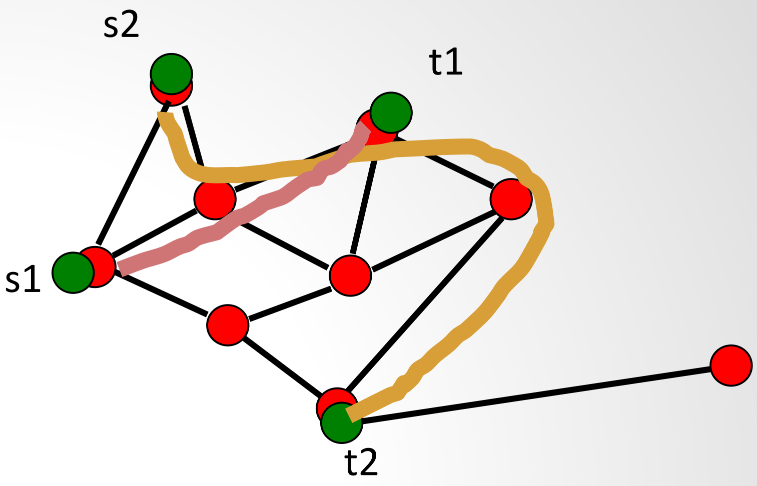
- Most simple: Minimum Linear Arrangement without capacities

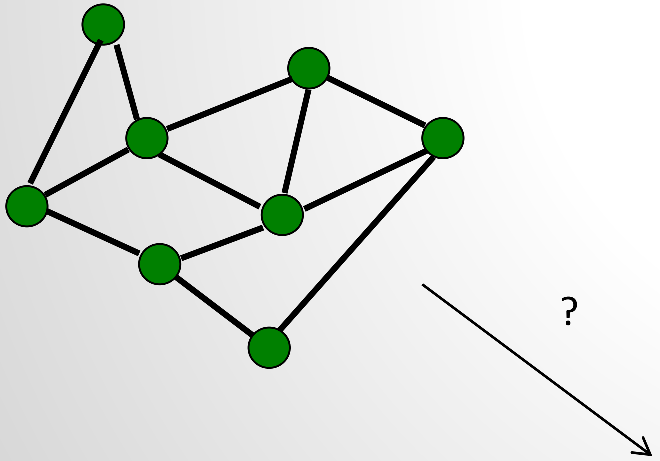


Let’s Exploit Allocation Flexibilities to Maximize Utilization

Start simple: exploit flexible routing between given VMs

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





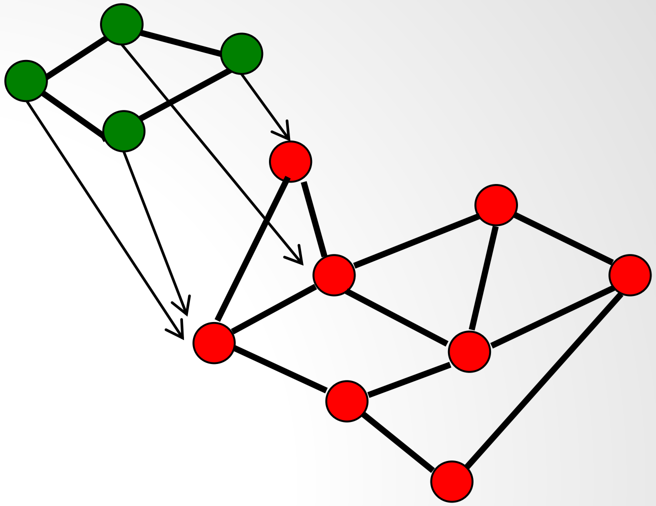
Forget about paths: exploit VM placement flexibilities!

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

That's all Folks!

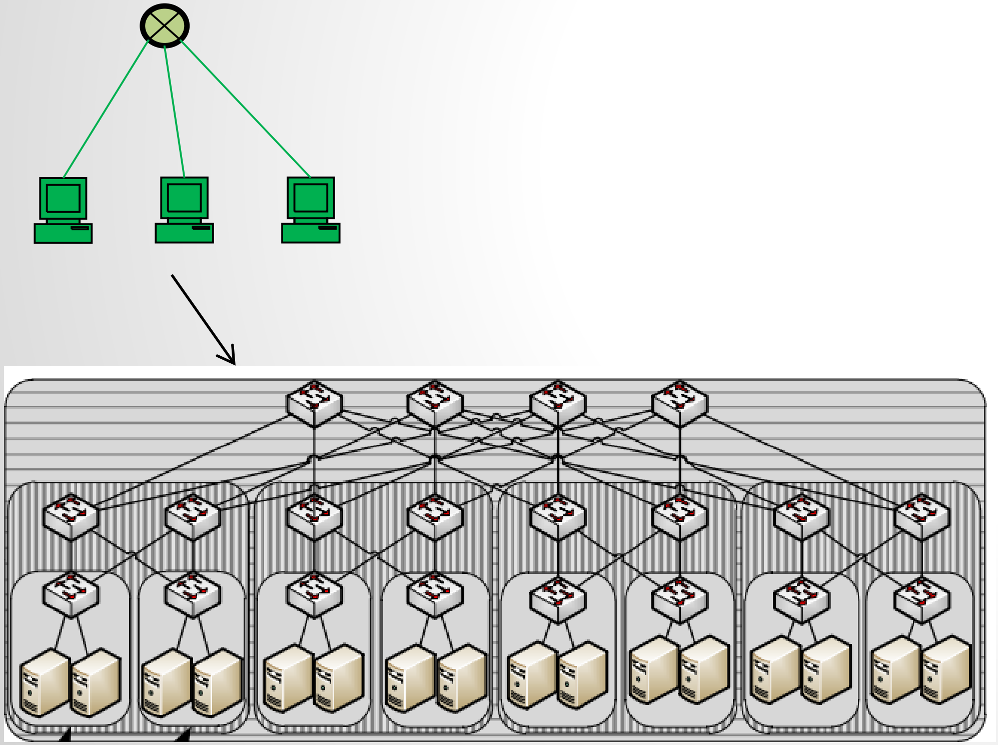
Wait a minute!
These problems need to be solved!
And they often can, even with guarantees.

Theory vs Practice



Goal in theory:

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

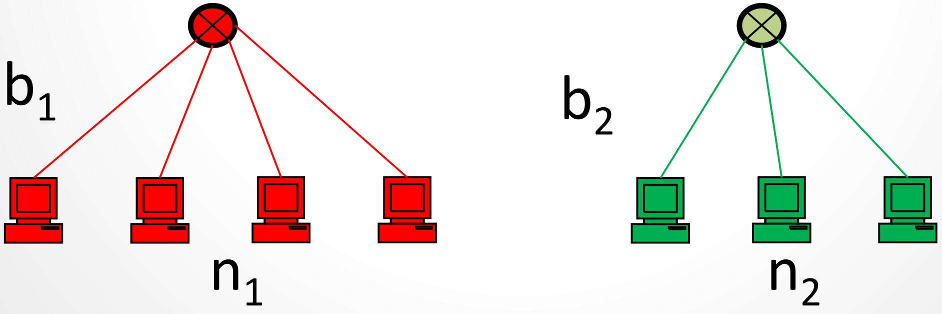


Reality:

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

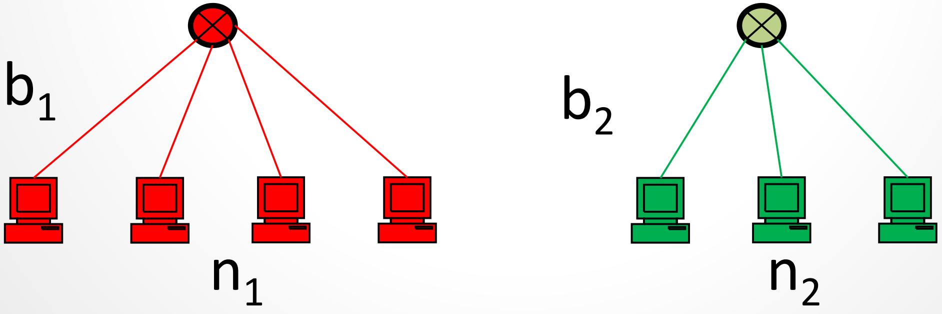
Virtual Clusters

- 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



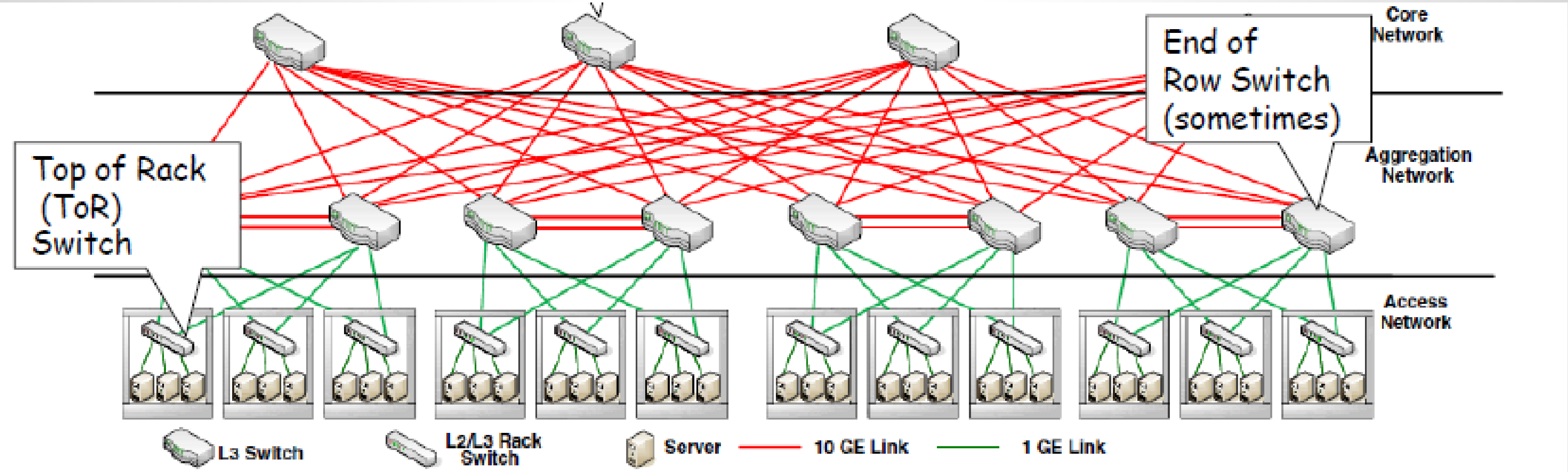
Virtual Clusters

- 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?

Fat-Tree Networks in Reality

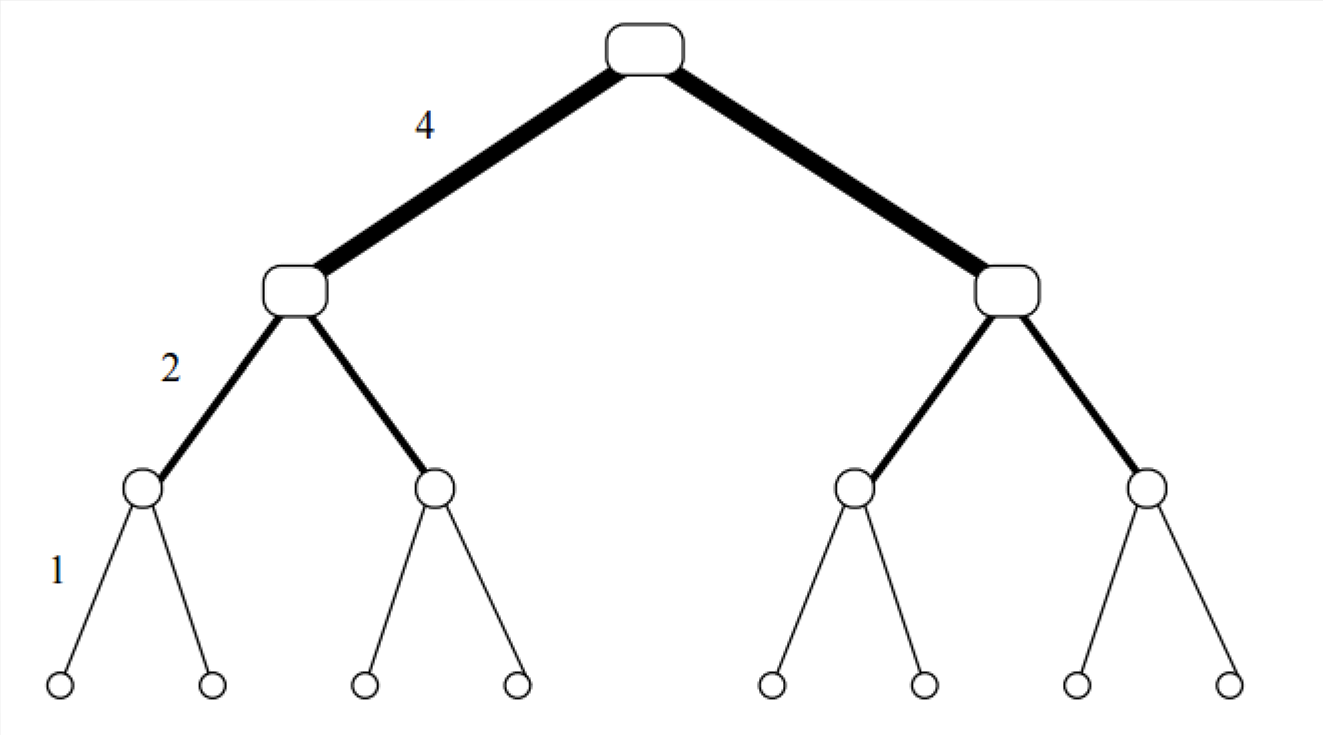


Top of Rack (ToR) Switch

End of Row Switch (sometimes)

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,

A Typical Datacenter Topology

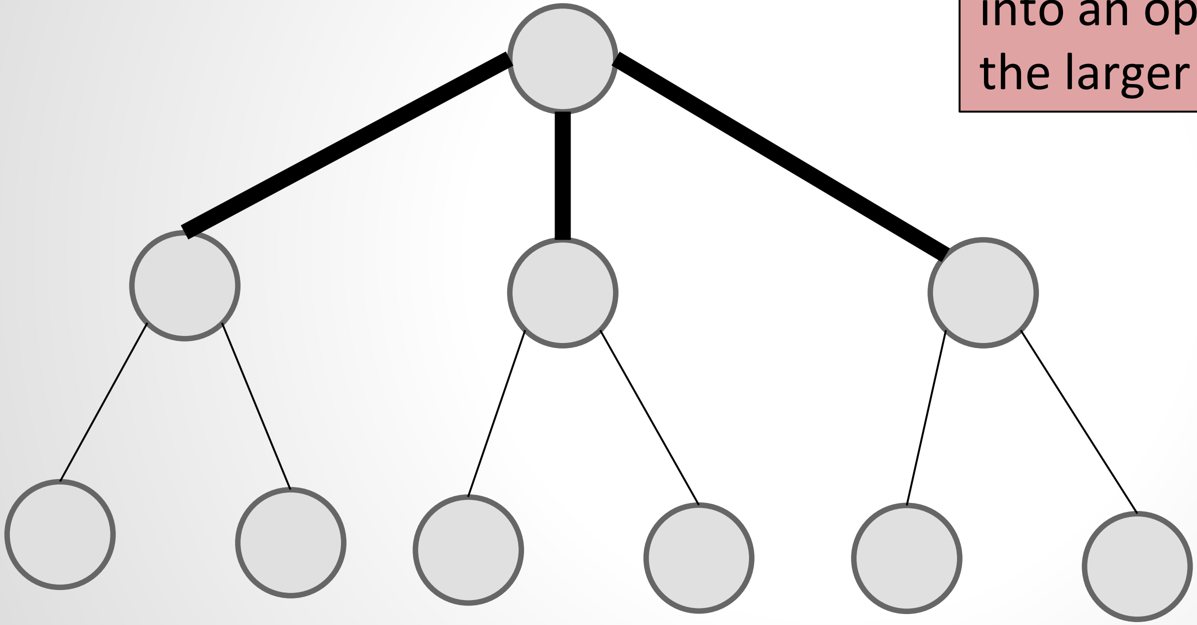


But due to ECMP, often ok to think of it like this.

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

- Example: dynamic programming

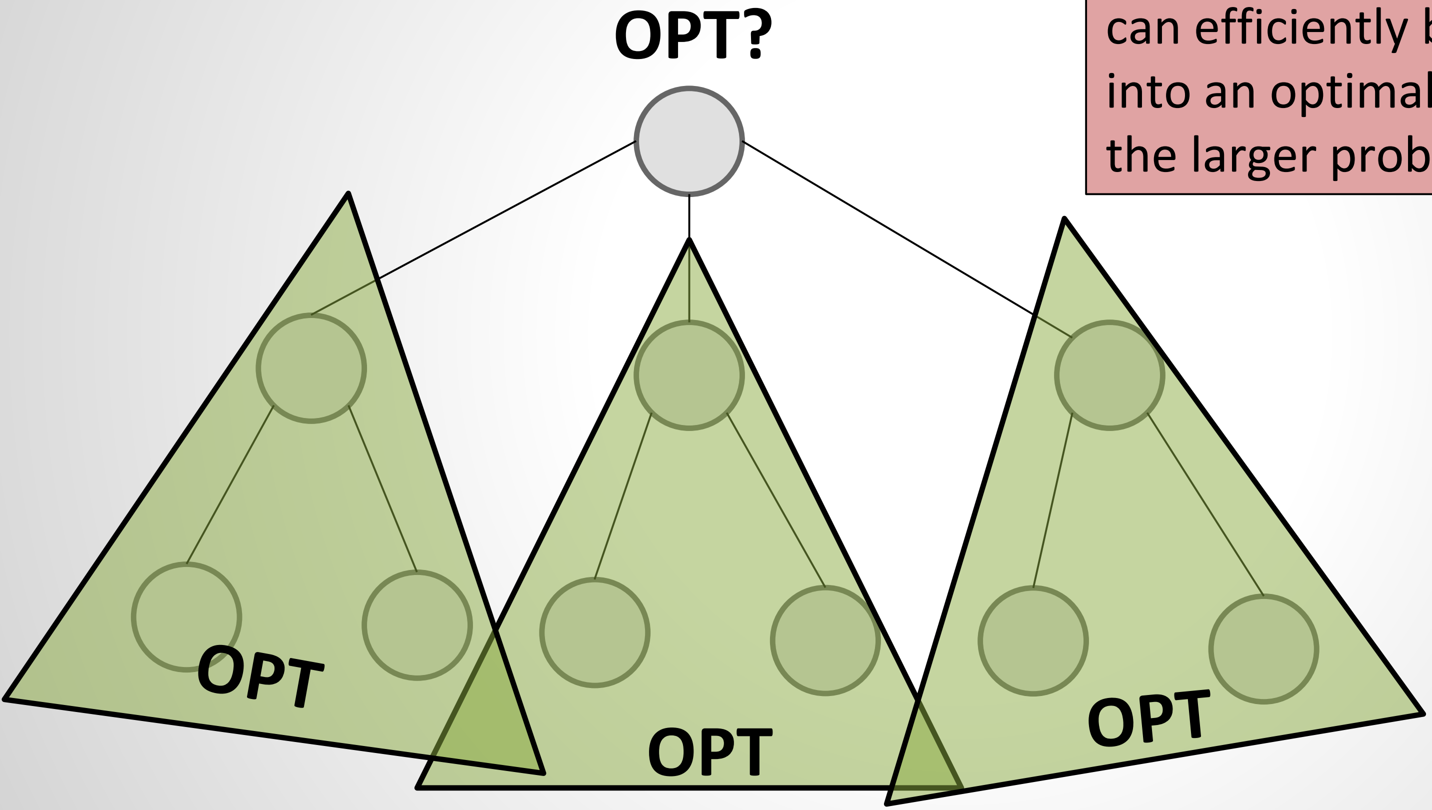
Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!



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

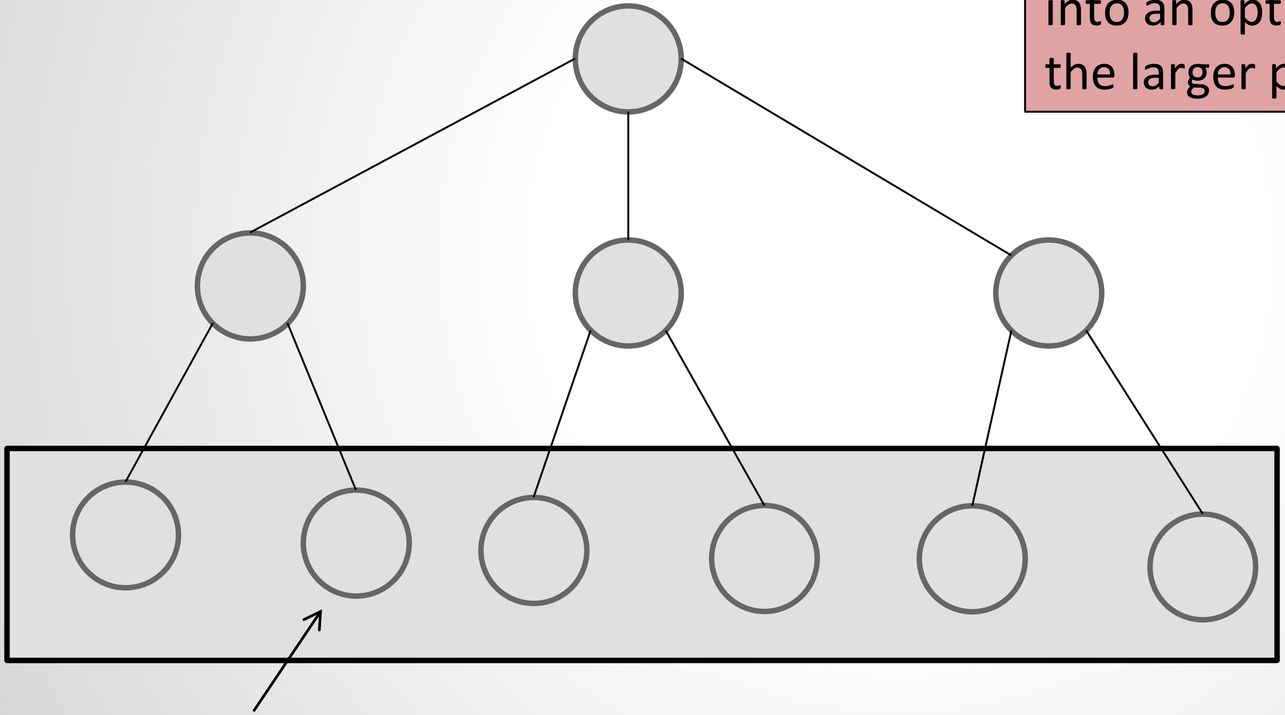
- Example: dynamic programming

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!



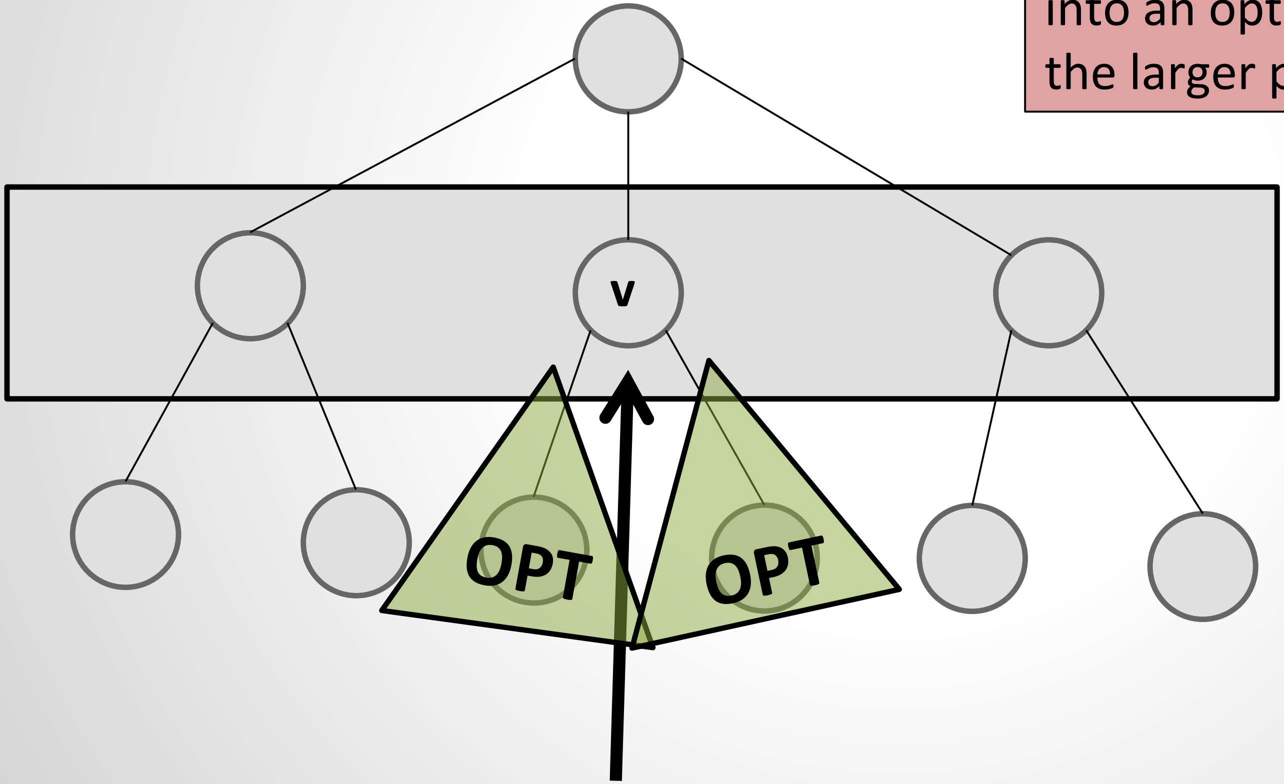
t = 0: solve leaves!

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!



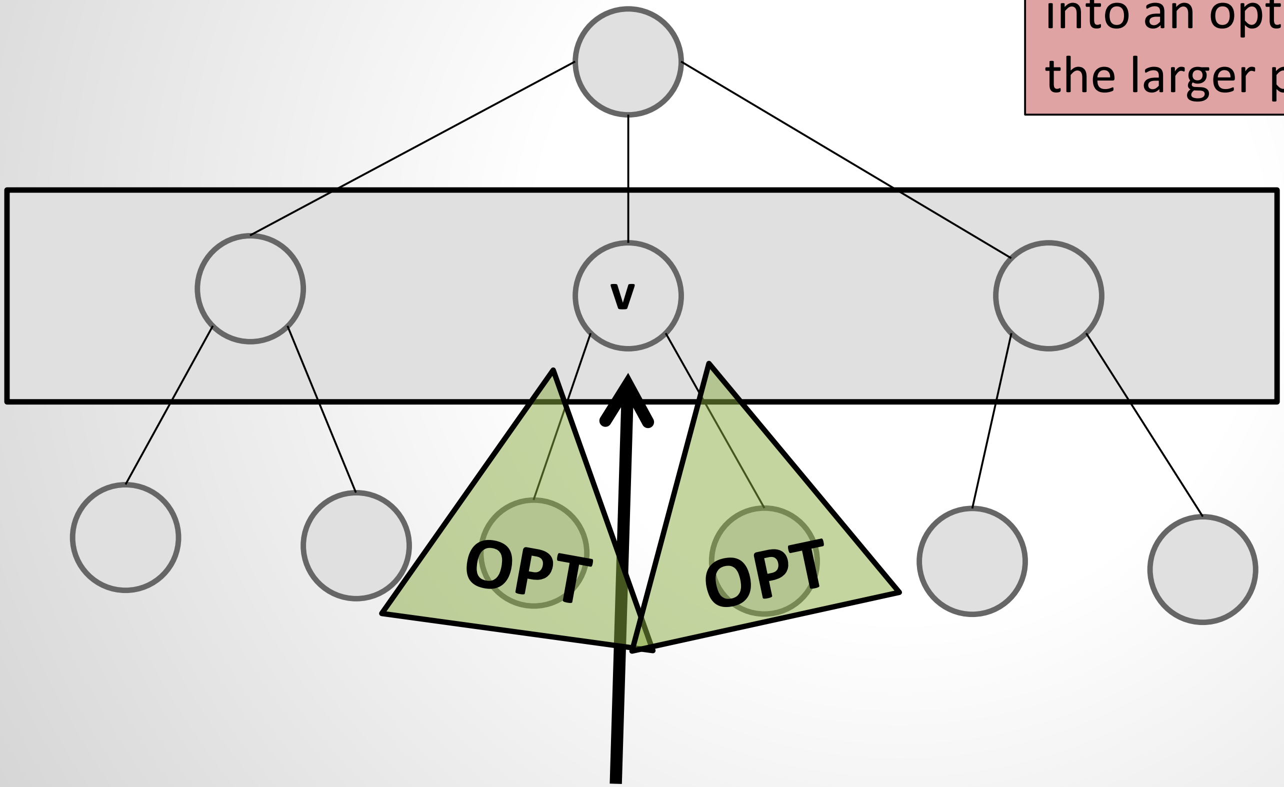
t = 1: solve height 1!

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

+ 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!



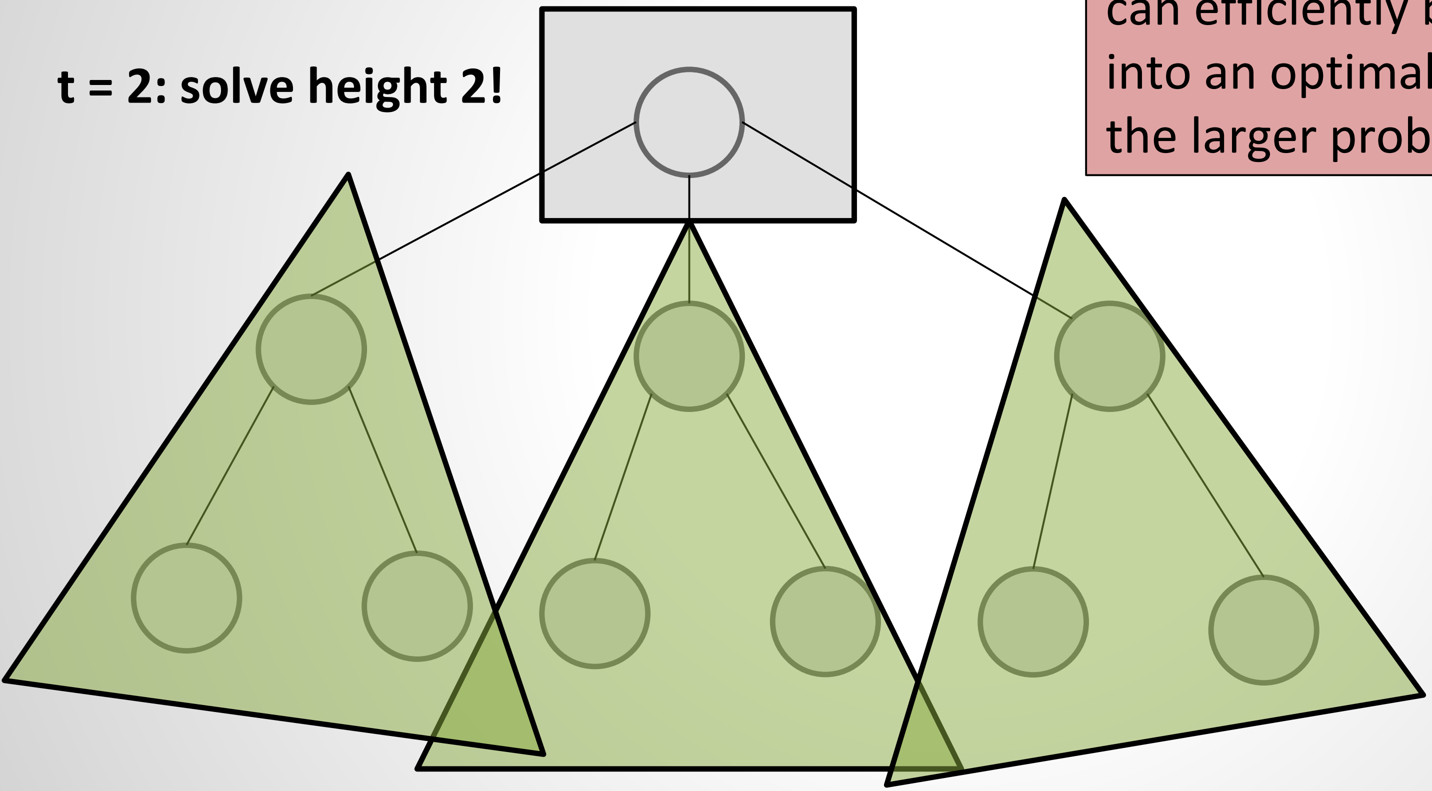
t = 1: solve height 1!

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

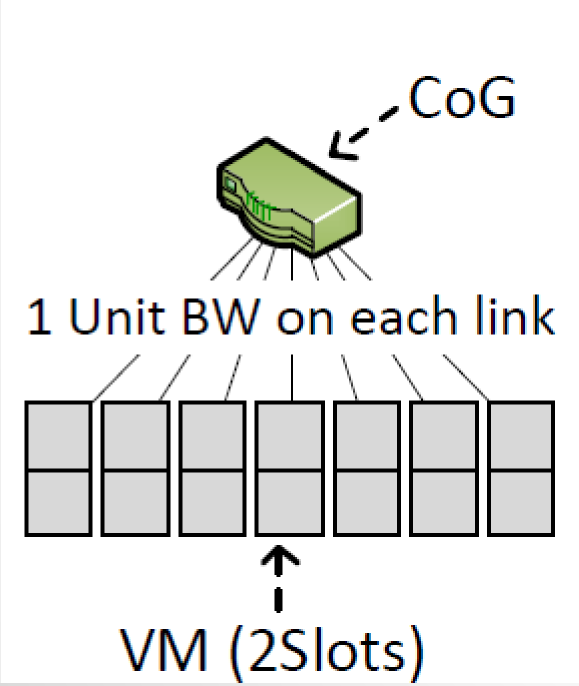
t = 2: solve height 2!

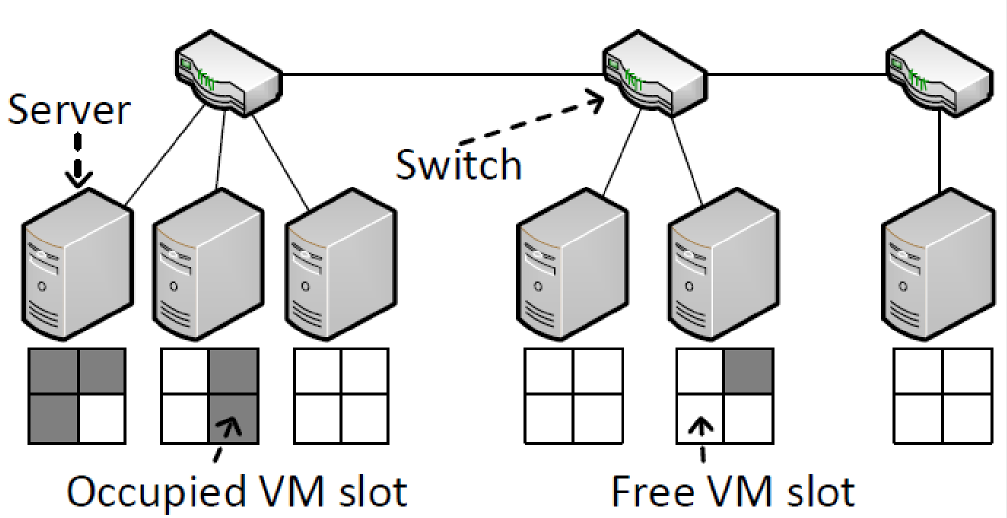


Dynamic Program = optimal solutions for subproblems can efficiently be combined into an optimal solution for the larger problem!

How to embed a Virtual Cluster in a General Graph?

How to embed?





Host Graph

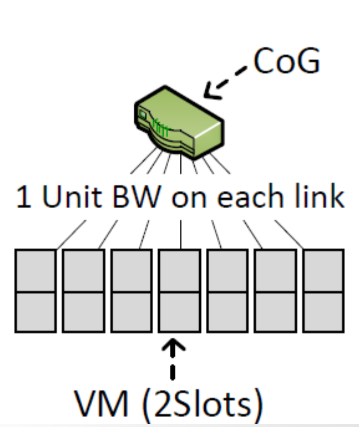
Guest Graph

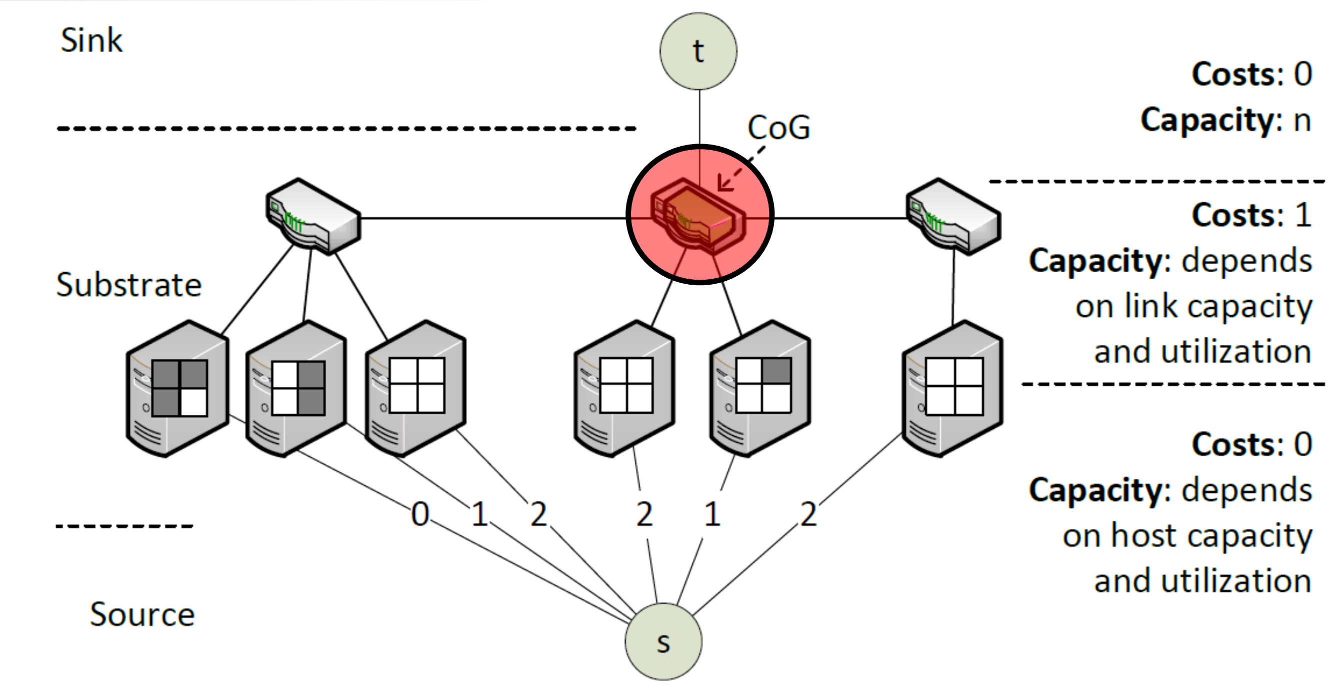
How to embed a Virtual Cluster in a General Graph?

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)





Costs: 0
Capacity: n

Costs: 1
Capacity: depends on link capacity and utilization

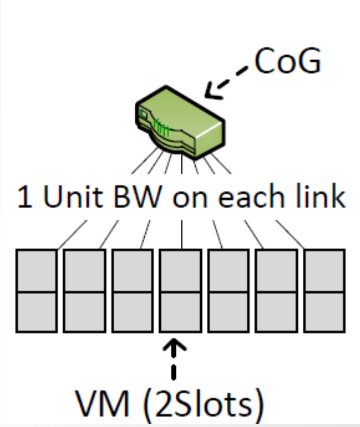
Costs: 0
Capacity: depends on host capacity and utilization

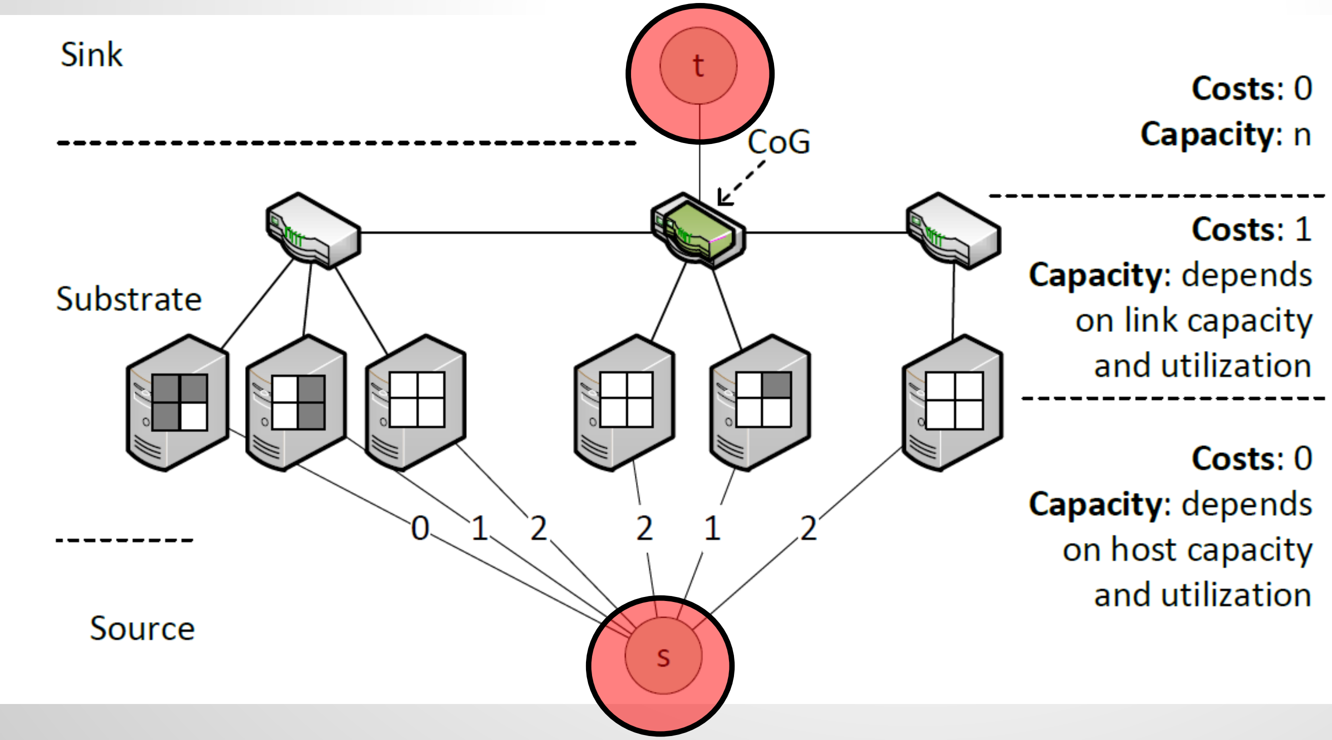
How to embed a Virtual Cluster in a General Graph?

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)





Costs: 0
Capacity: n

Costs: 1
Capacity: depends on link capacity and utilization

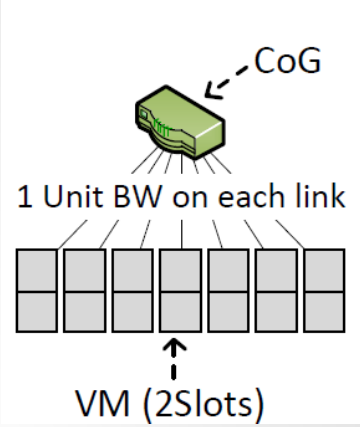
Costs: 0
Capacity: depends on host capacity and utilization

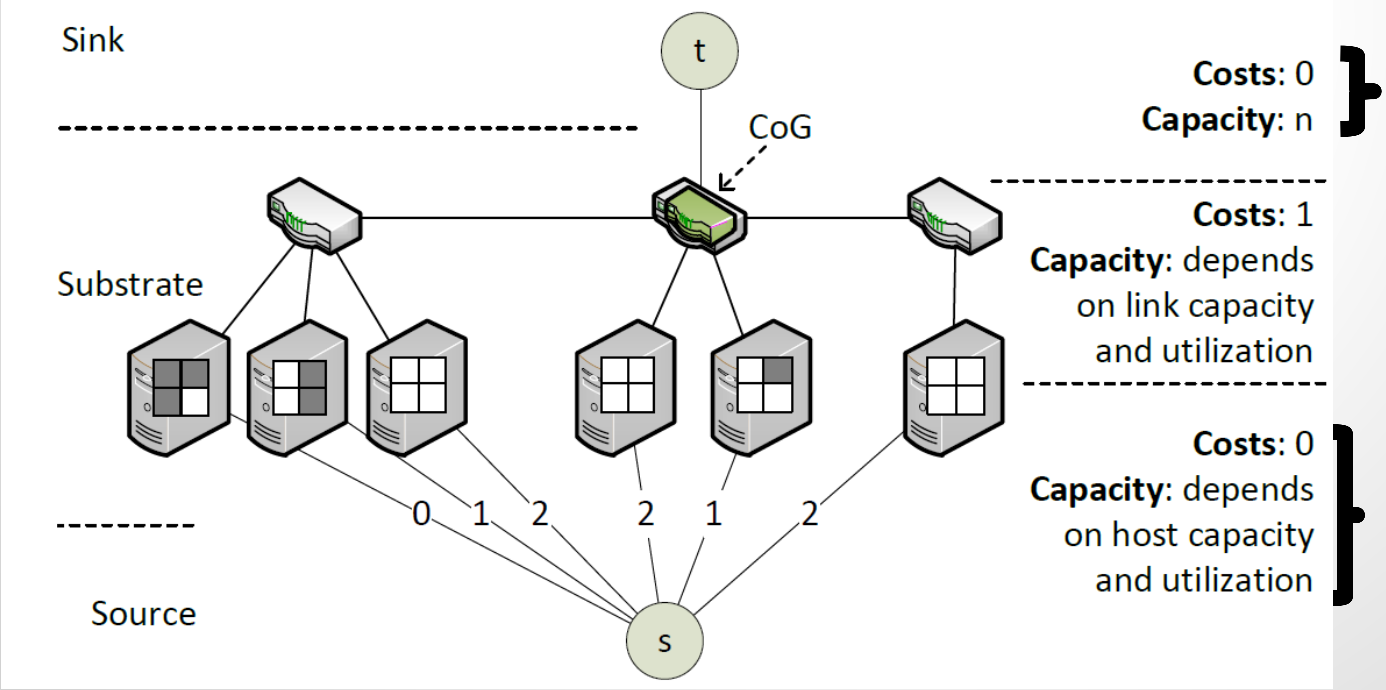
How to embed a Virtual Cluster in a General Graph?

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)





Costs: 0
Capacity: n

Costs: 1
Capacity: depends on link capacity and utilization

Costs: 0
Capacity: depends on host capacity and utilization

enough to embed n VMs

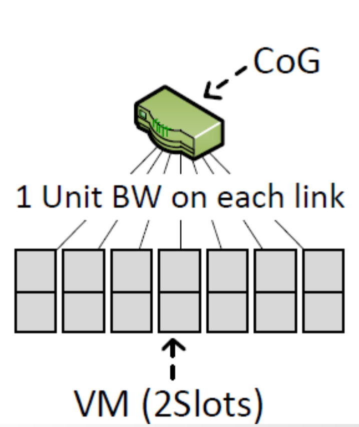
capacity = floor(available resources / unit demand)

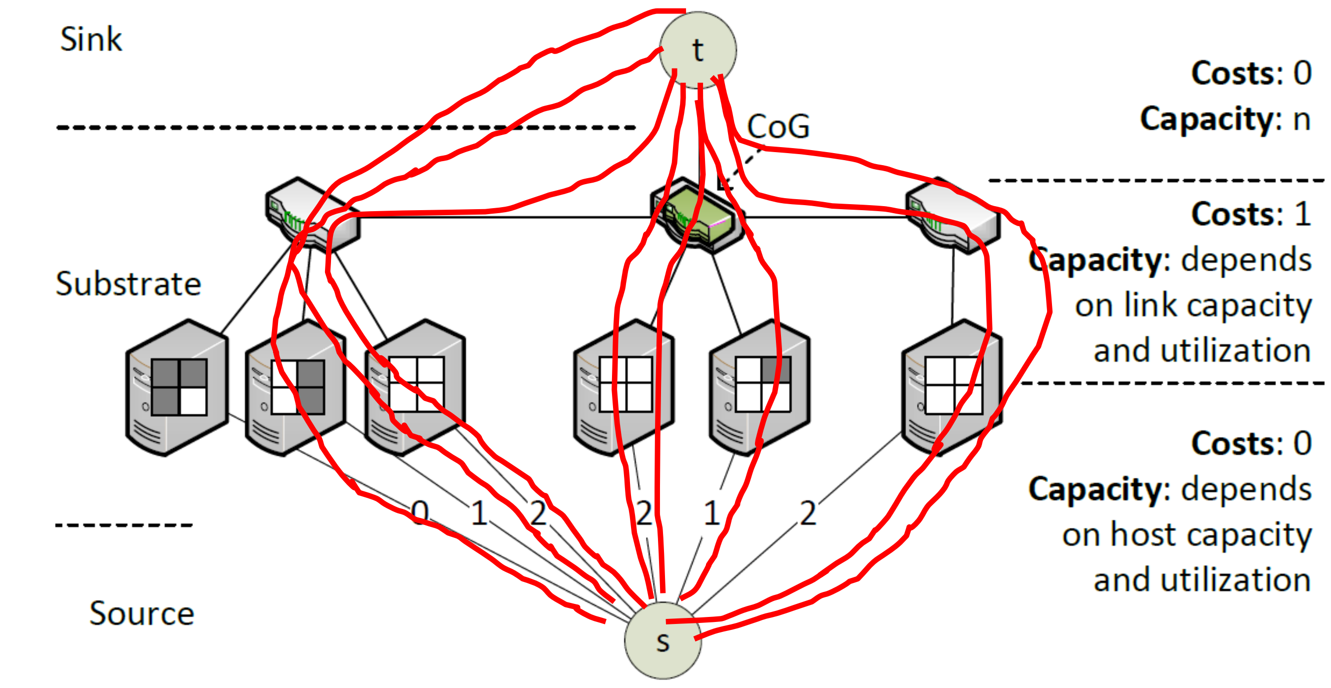
How to embed a Virtual Cluster in a General Graph?

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)





Costs: 0
Capacity: n

Costs: 1
Capacity: depends on link capacity and utilization

Costs: 0
Capacity: depends on host capacity and utilization

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

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.

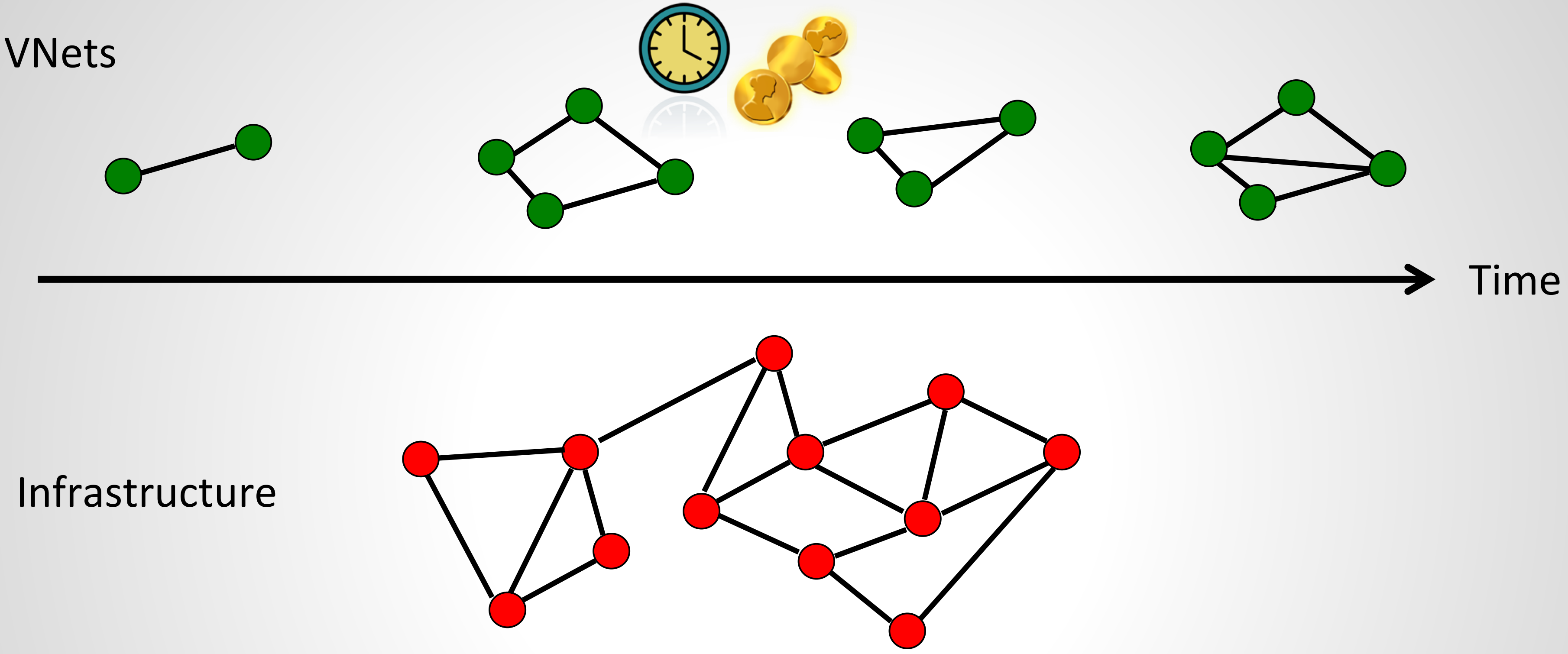
Competitive Analysis

.

Competitive ratio ρ: max over all possible sequences σ

ρ = Cost(ON)/Cost(OFF)

Online Access Control (1)



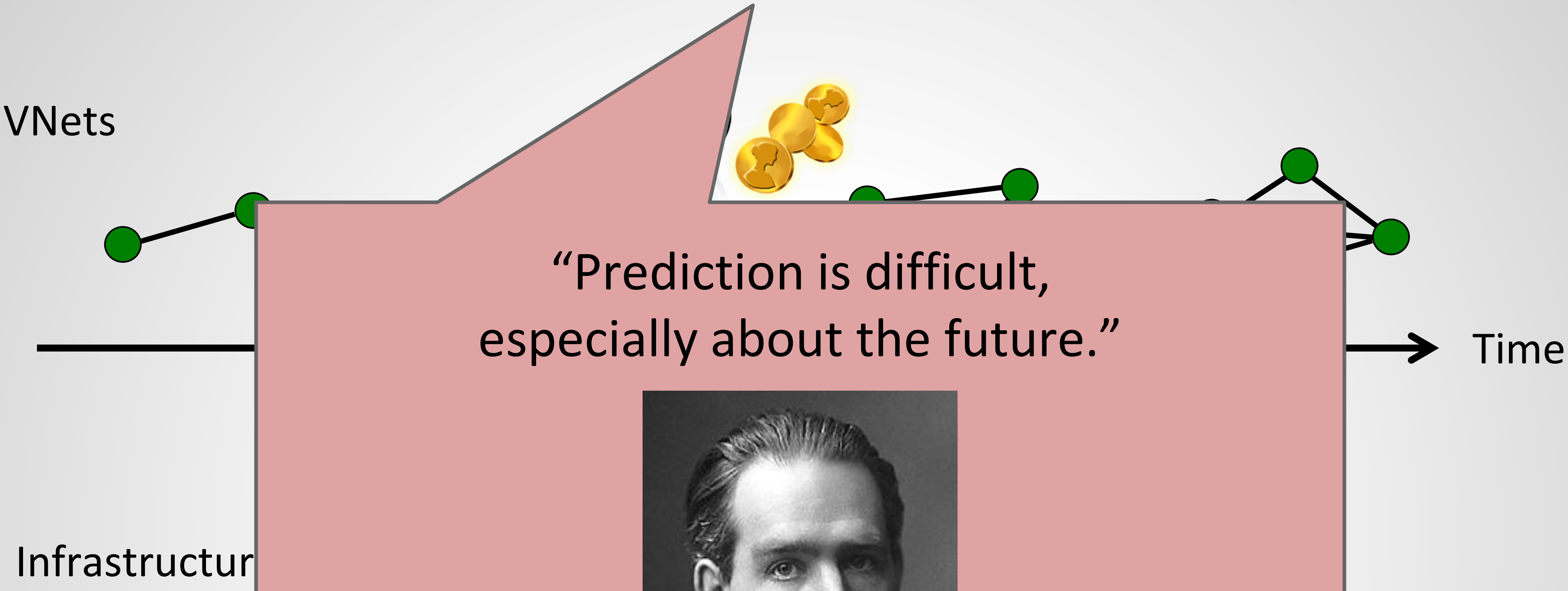
VNets

Time

Infrastructure

- □ 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)



"Prediction is difficult,
especially about the future."

Nils Bohr

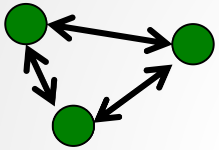
- ❑ Assume:
- ❑ Differen
- ❑ Price and duration
- ❑ Which ones to accept?
- ❑ Online Primal-Dual Framework (Buchbinder and Naor)

Online Access Control (2)

❑ Traffic models

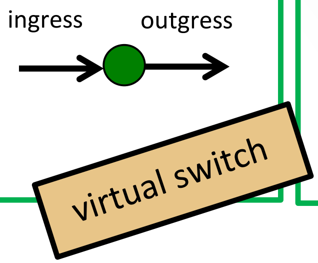
Customer Pipe

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



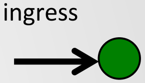
Hose Model

Per VM bandwidth: polytope of traffic matrices.



Aggregate Ingress

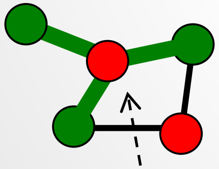
Only ingress specified: e.g., support multicast etc.



❑ Routing models

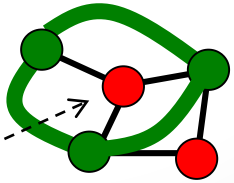
Tree

Steiner tree embedding



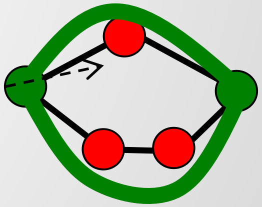
Single Path

Unsplittable paths



Multi-Path

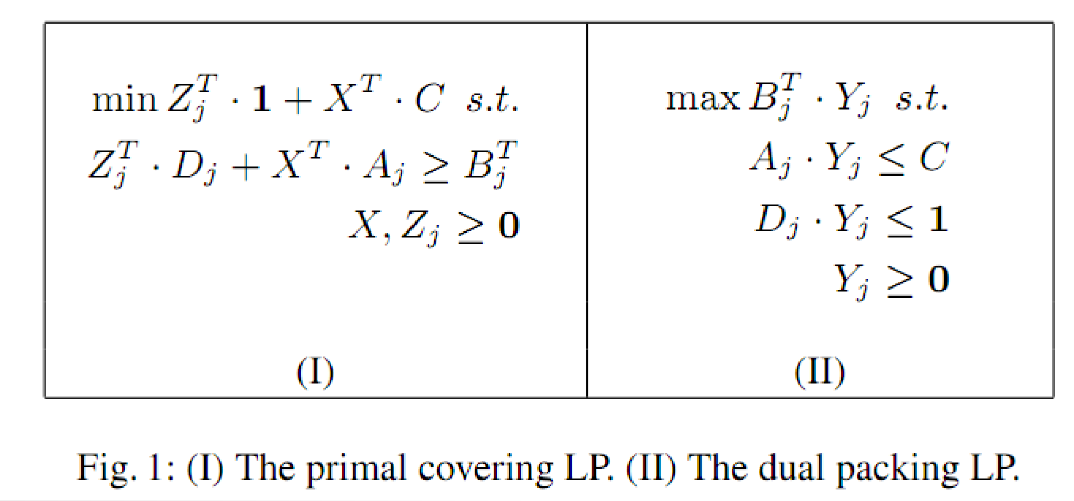
Splittable paths (more capacity)



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

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.

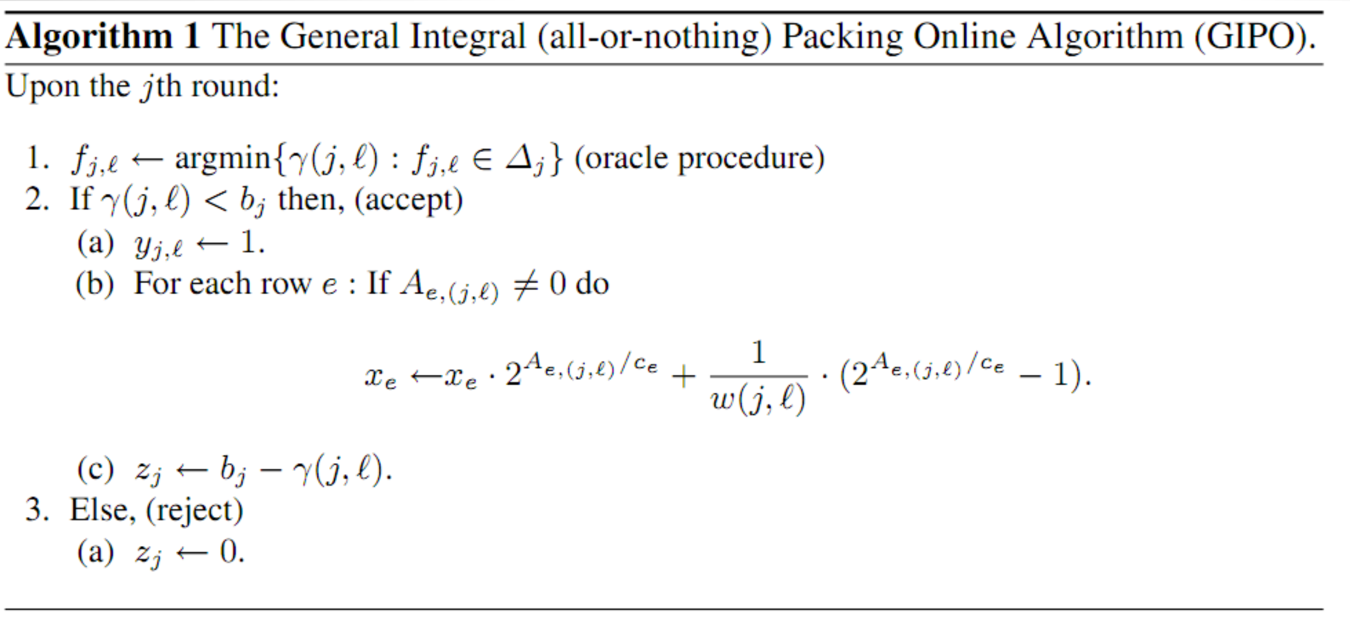
Competitive Analysis

Does not know t'>t.

Competitive ratio:

r = Cost(ON)/Cost(OFF)

Algorithm

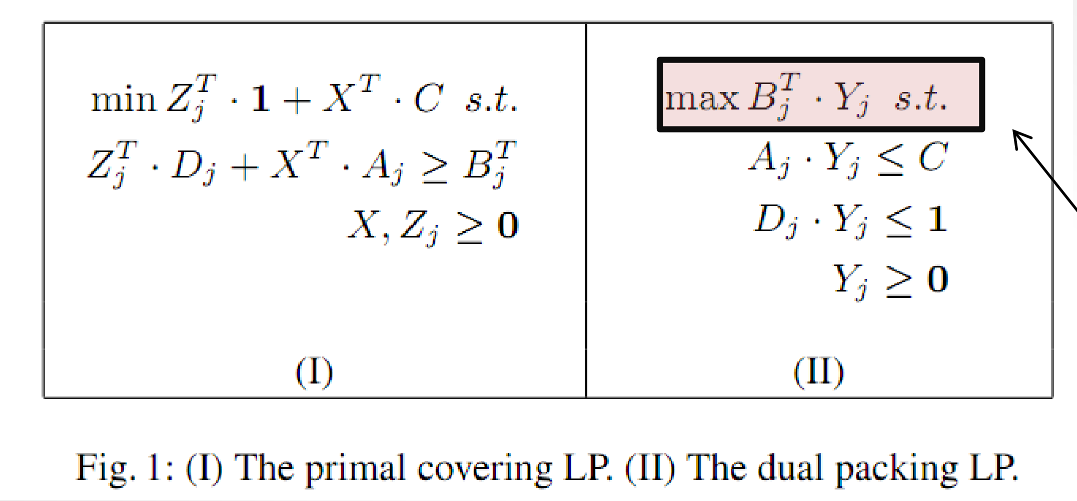


Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the $j$th 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)



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

(I)

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

(II)

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

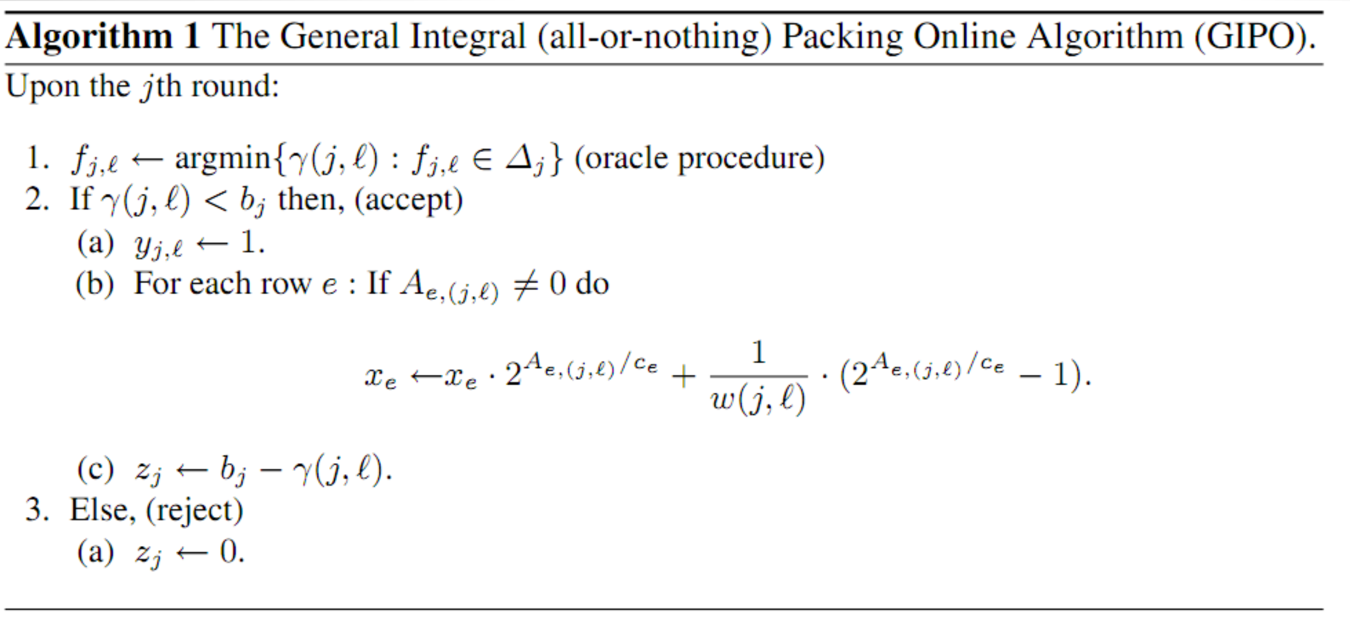
Formulate the packing (dual) LP: Maximize profit

(Note: dynamic LP!)

Competitive Analysis

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

Algorithm

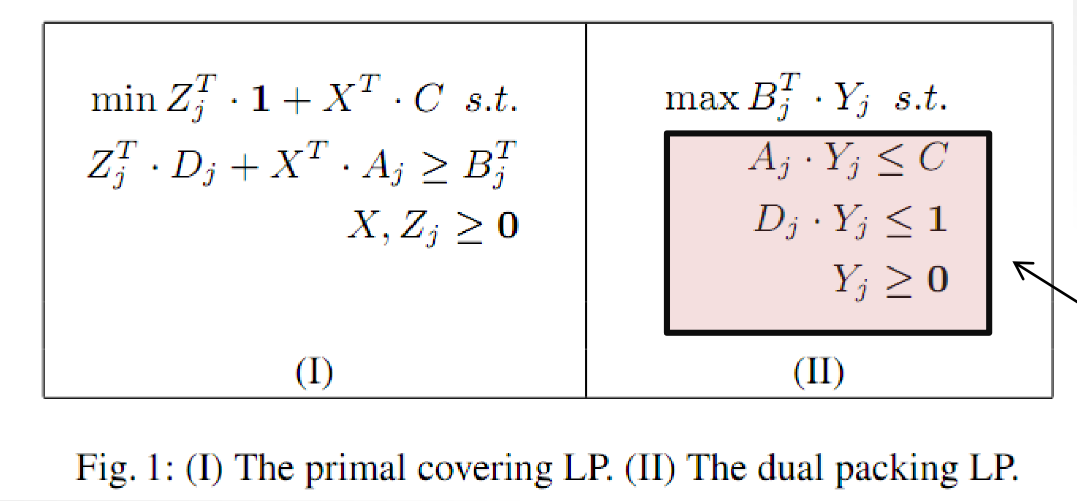


Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the jth round:
  1. $f_{j,\ell} \leftarrow \mathrm{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (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)



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

(I)

$$\max B_j^T \cdot Y_j \ \ s.t.$$
$$\boxed{\begin{aligned} A_j \cdot Y_j &\leq C \\ D_j \cdot Y_j &\leq \mathbf{1} \\ Y_j &\geq \mathbf{0} \end{aligned}}$$

(II)

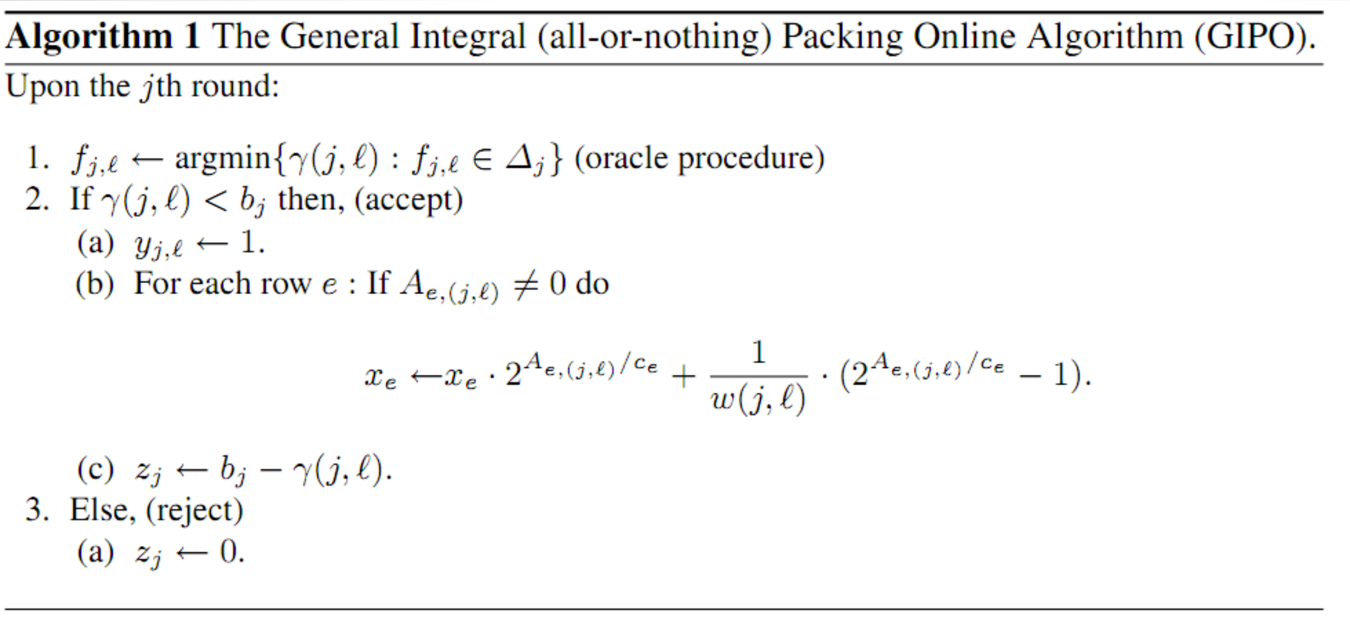
Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

s.t. constraints

Competitive Analysis

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

Algorithm

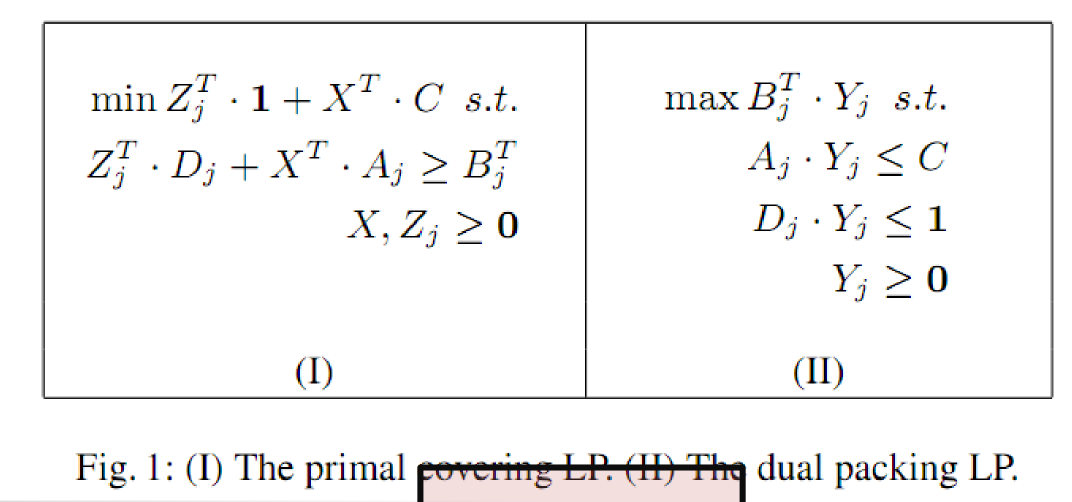


Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the $j$th 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)



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

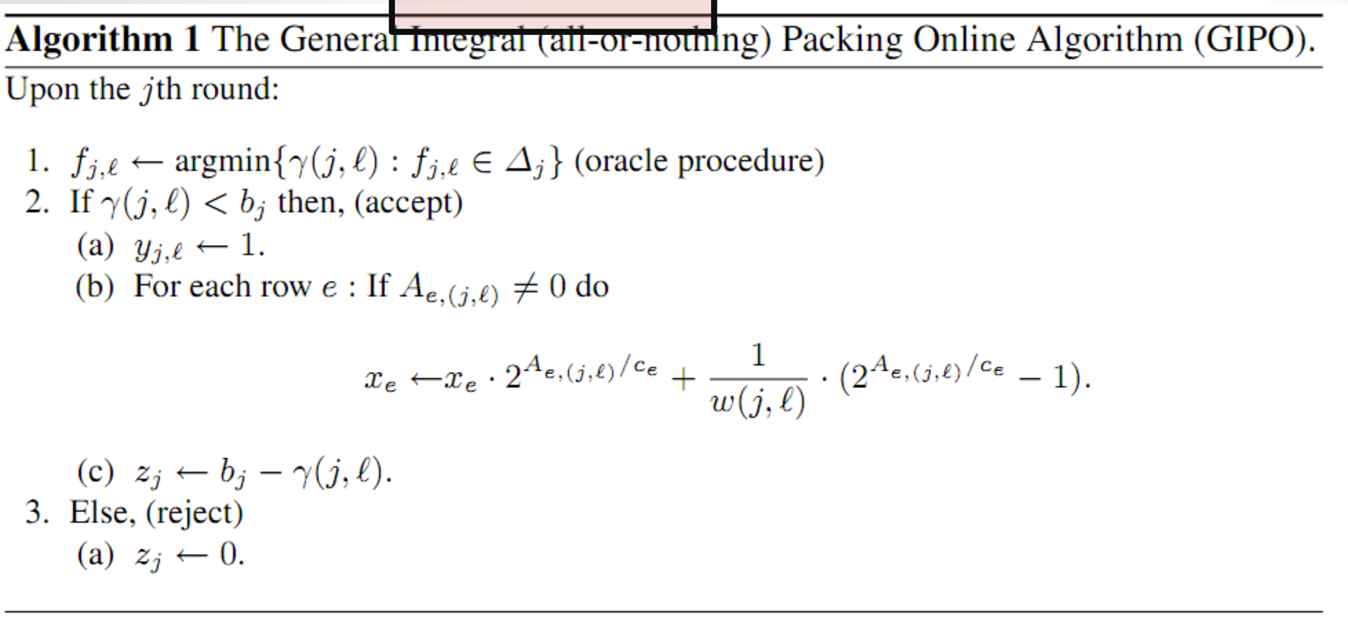
Fig. 1: (I) The primal covering LP. (II) The dual packing LP.

Competitive Analysis

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

primal-dual framework

Algorithm

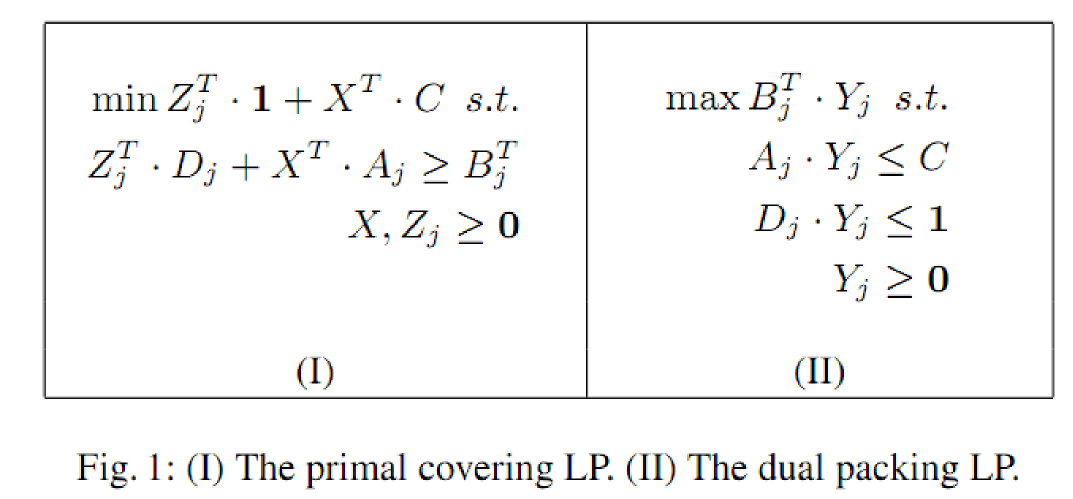


Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the $j$th round:

1. $f_{j,\ell} \leftarrow \operatorname{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (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



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

(I)

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

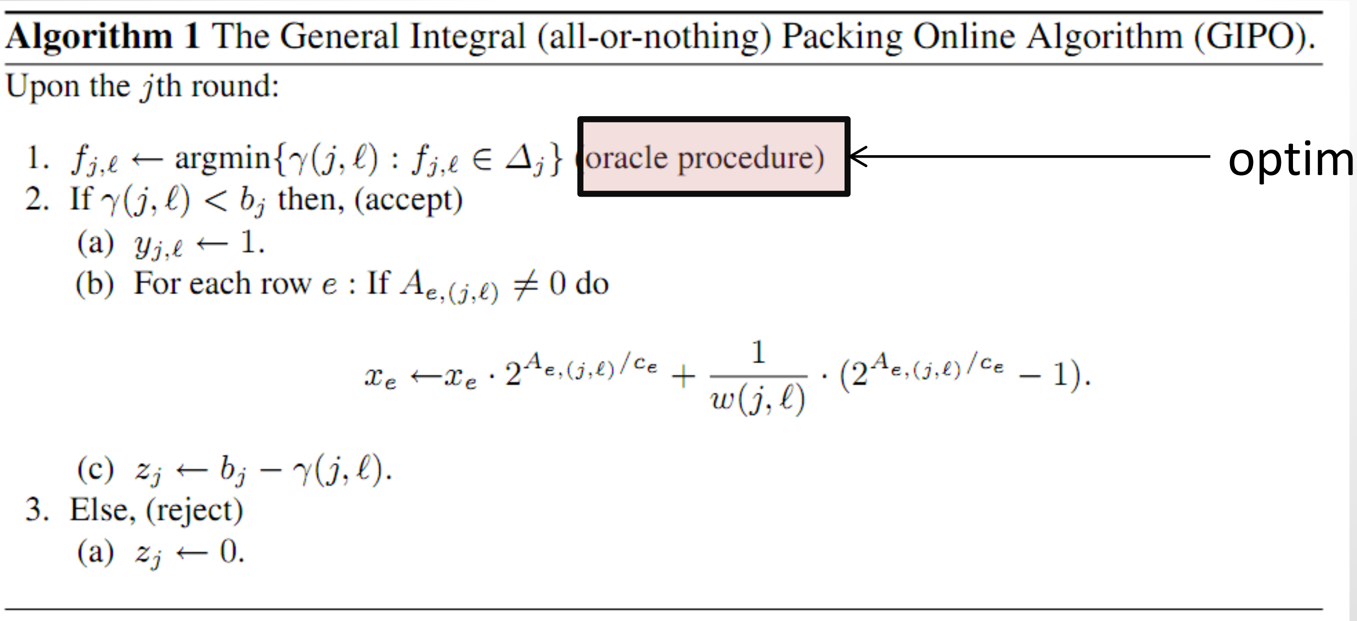
(II)

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



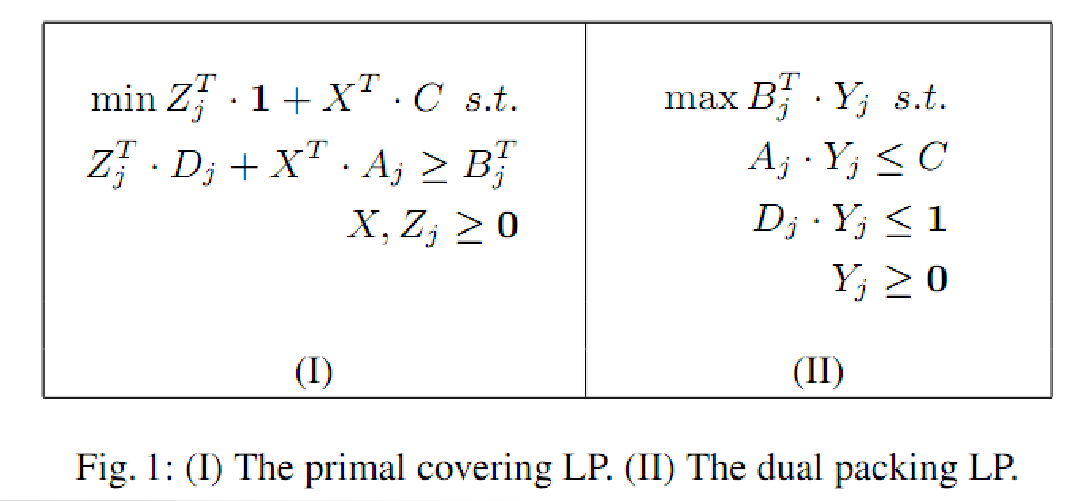
Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the $j$th 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) $z_j \leftarrow 0$.
```

optimal embedding!

Online Access Control (3)

Primal and Dual



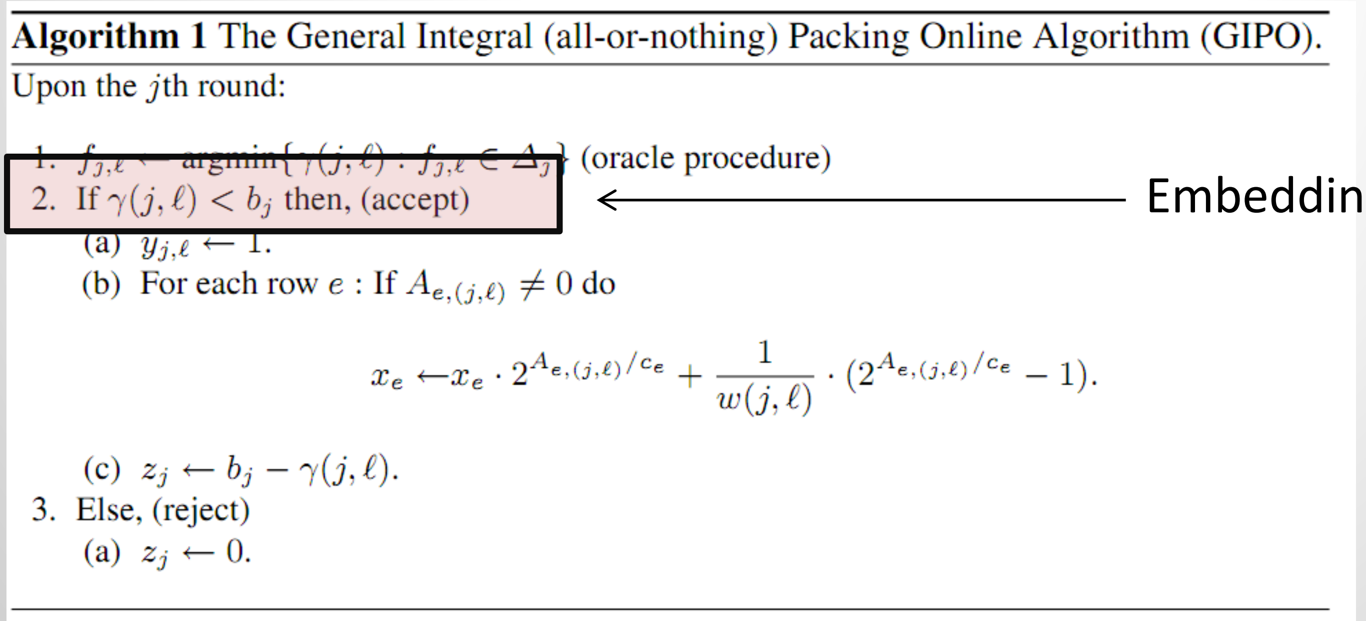
$\min Z_j^T \cdot \mathbf{1} + X^T \cdot C \;\; s.t.$ $Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T$ $X, Z_j \geq \mathbf{0}$	$\max B_j^T \cdot Y_j \;\; s.t.$ $A_j \cdot Y_j \leq C$ $D_j \cdot Y_j \leq \mathbf{1}$ $Y_j \geq \mathbf{0}$
(I)	(II)

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



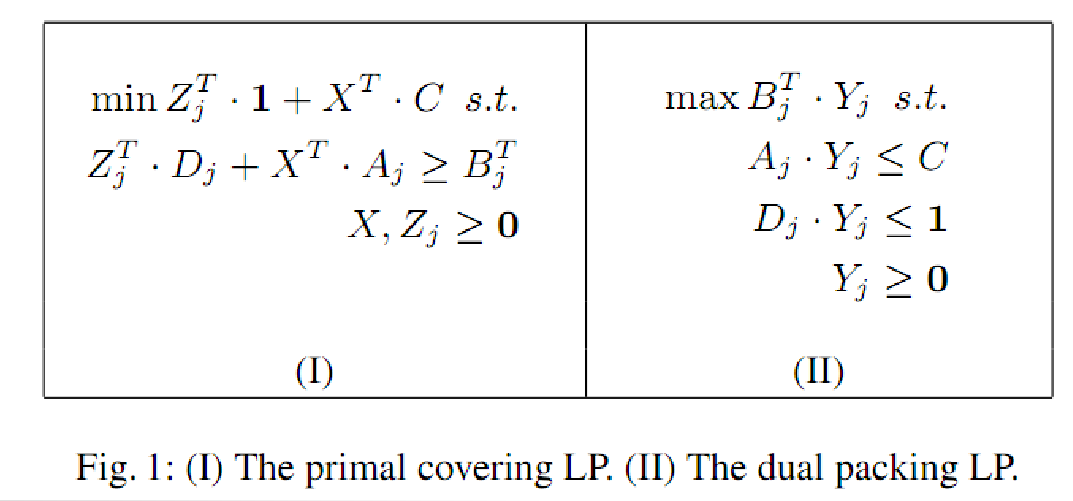
Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the $j$th round:
1. $f_{j,\ell} \leftarrow \arg\min\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (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



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

(I)

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

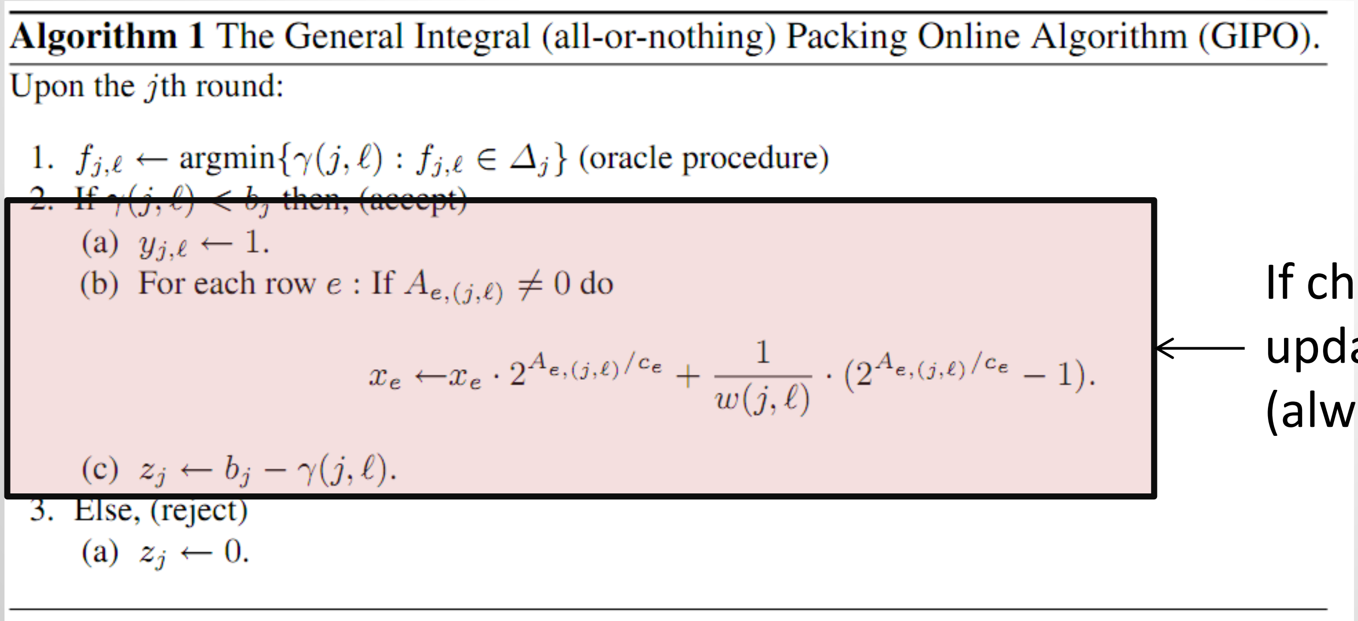
(II)

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

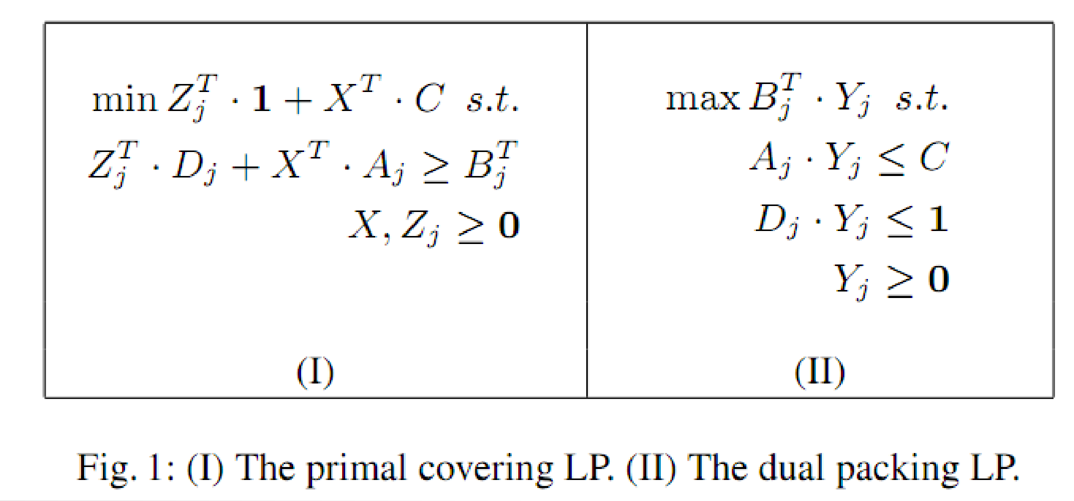


```
Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).
Upon the $j$th round:
  1. $f_{j,\ell} \leftarrow \text{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
  2. If $\gamma(j,\ell) < b_j$ then, (accept)
      (a) $y_{j,\ell} \leftarrow 1$.
      (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)



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

(I)

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

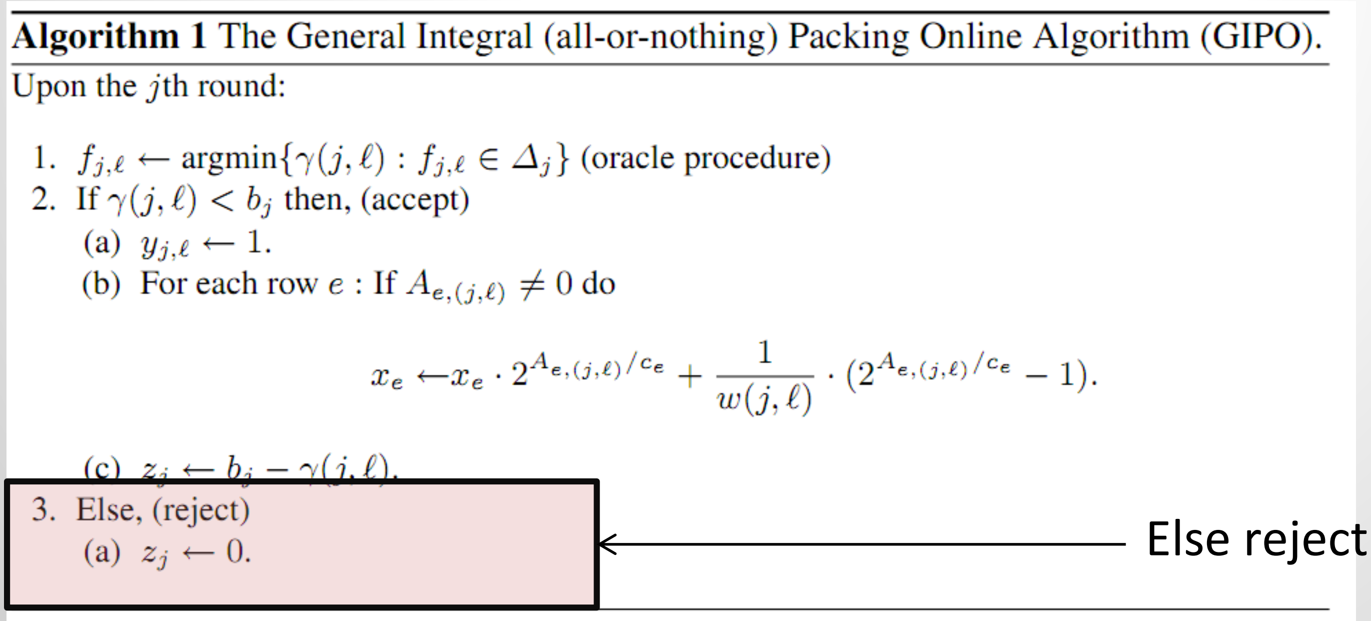
(II)

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



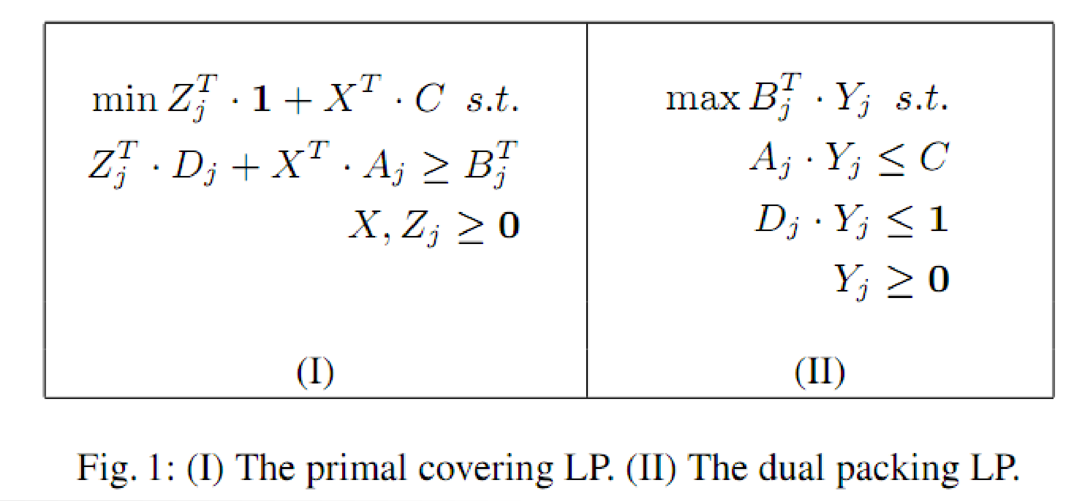
Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the $j$th round:
  1. $f_{j,\ell} \leftarrow \operatorname{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (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

Online Access Control (3)

Primal and Dual



$$\begin{aligned} \min\ & Z_j^T \cdot \mathbf{1} + X^T \cdot C \quad s.t. \\ & Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T \\ & X, Z_j \geq \mathbf{0} \end{aligned}$$

(I)

$$\begin{aligned} \max\ & B_j^T \cdot Y_j \quad s.t. \\ & A_j \cdot Y_j \leq C \\ & D_j \cdot Y_j \leq \mathbf{1} \\ & Y_j \geq \mathbf{0} \end{aligned}$$

(II)

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

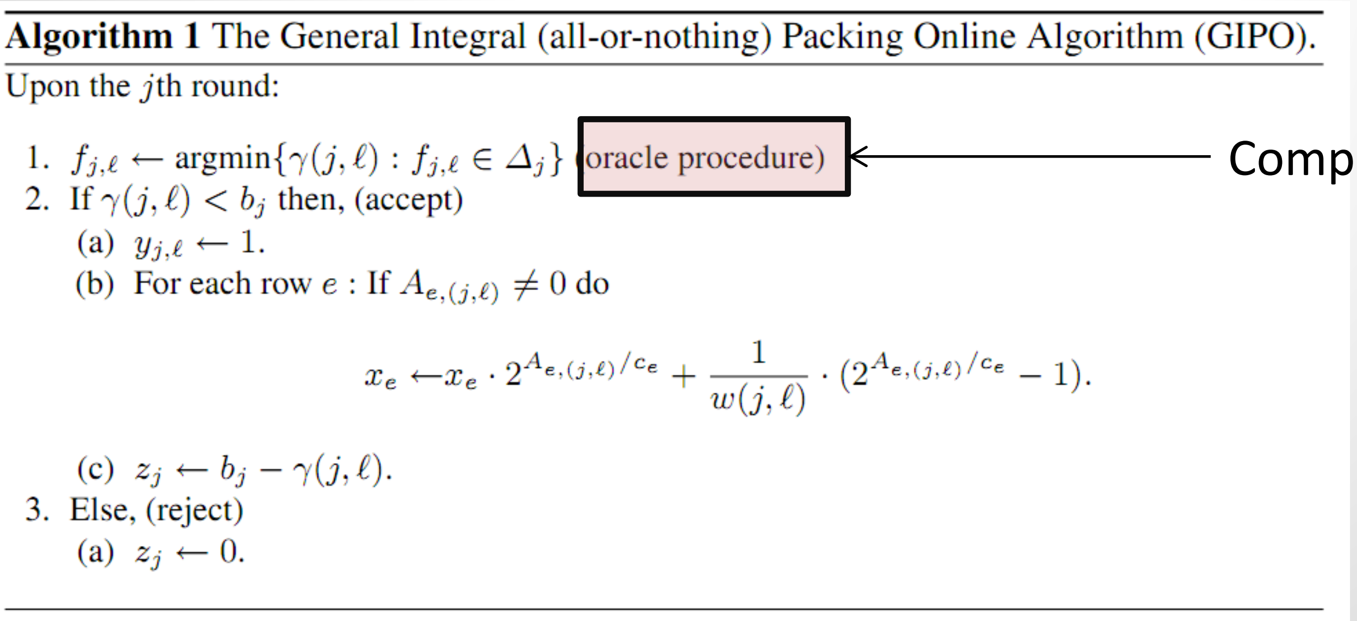
Competitive Analysis

Does not know t'>t.

Competitive ratio:

r = Cost(ON)/Cost(OFF)

Algorithm



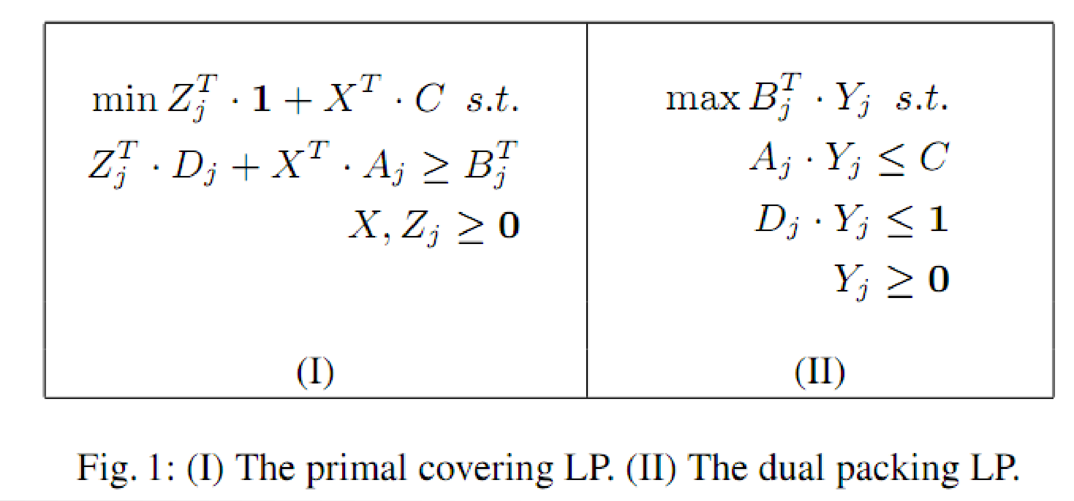
Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

```
Upon the jth round:
  1. $f_{j,\ell} \leftarrow \mathrm{argmin}\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (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!

Primal and Dual

Online Access Control (3)



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

(I)

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

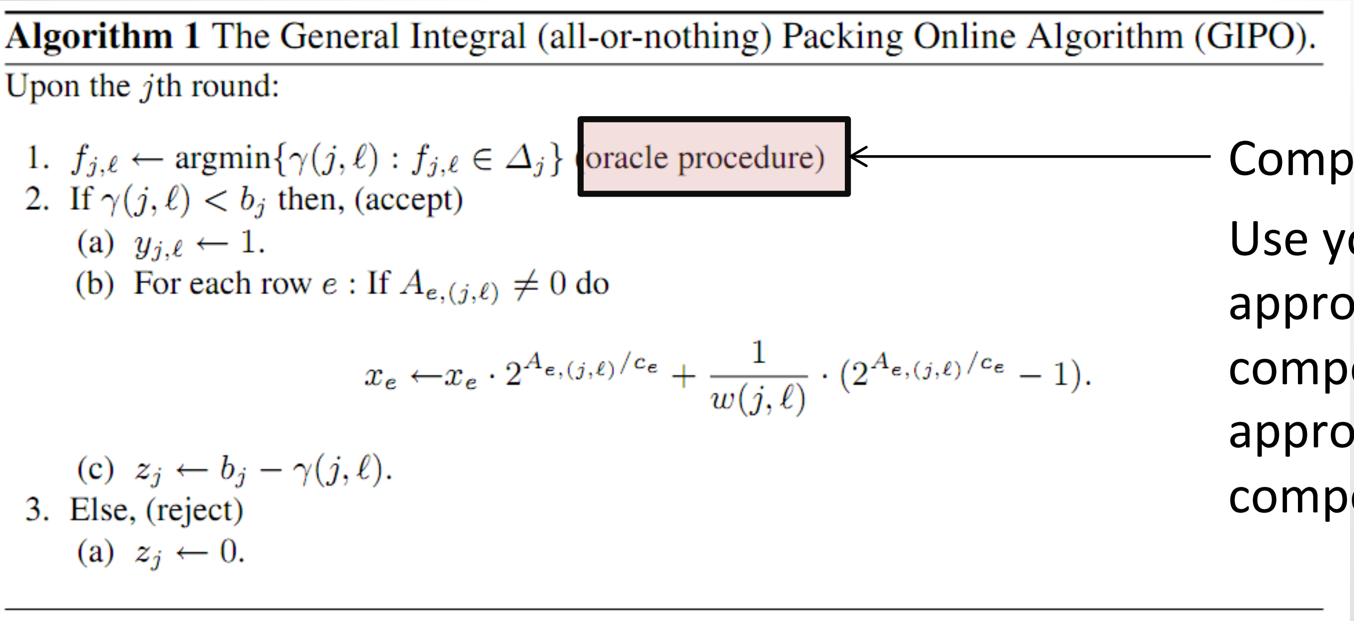
(II)

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 $j$th 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) $z_j \leftarrow 0$.
```

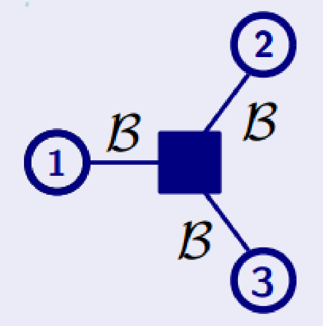
Computationally hard!

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

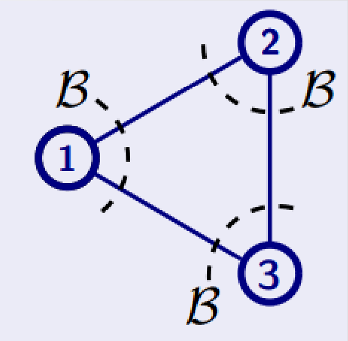
A Note on the Hose Model (1)

- Recall: Virtual Cluster Abstraction

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



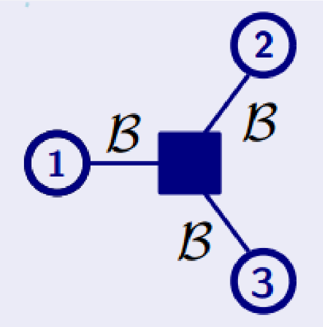
VS

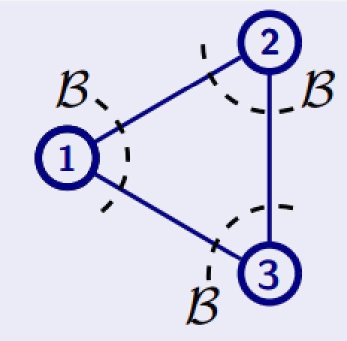


- If switch location unique
  - Polynomial-time algorithms: can try all locations…
  - … and then do our trick with the extra source.
  - What about Hose?

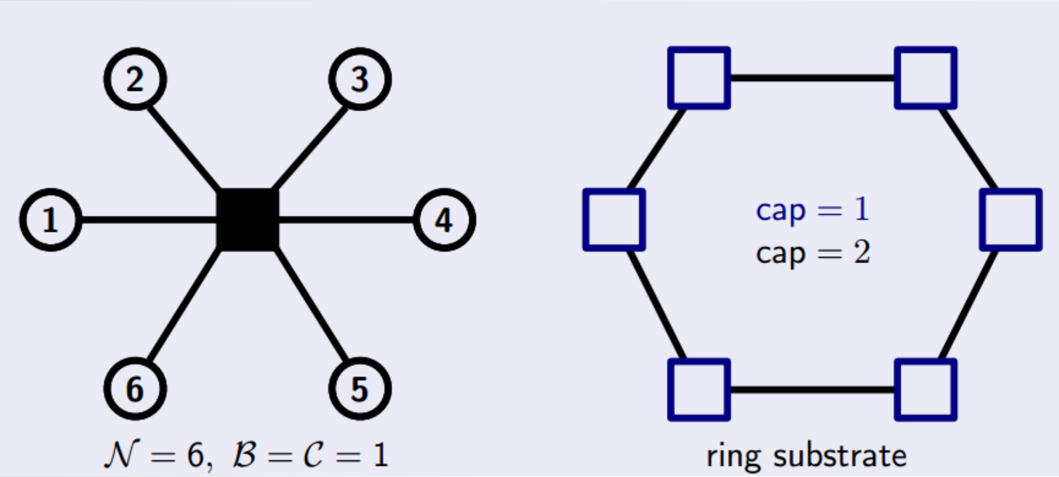
A Note on the Hose Model (2)

- ❑ 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





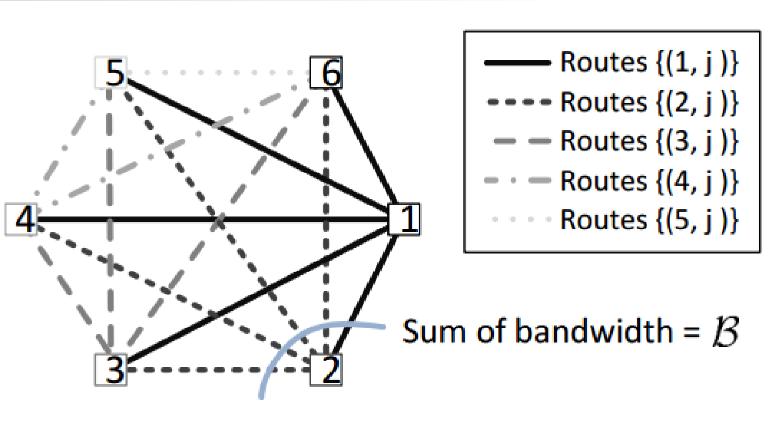
On the Benefit of Hose (1)



❑ VC: Compute and bandwidth one unit

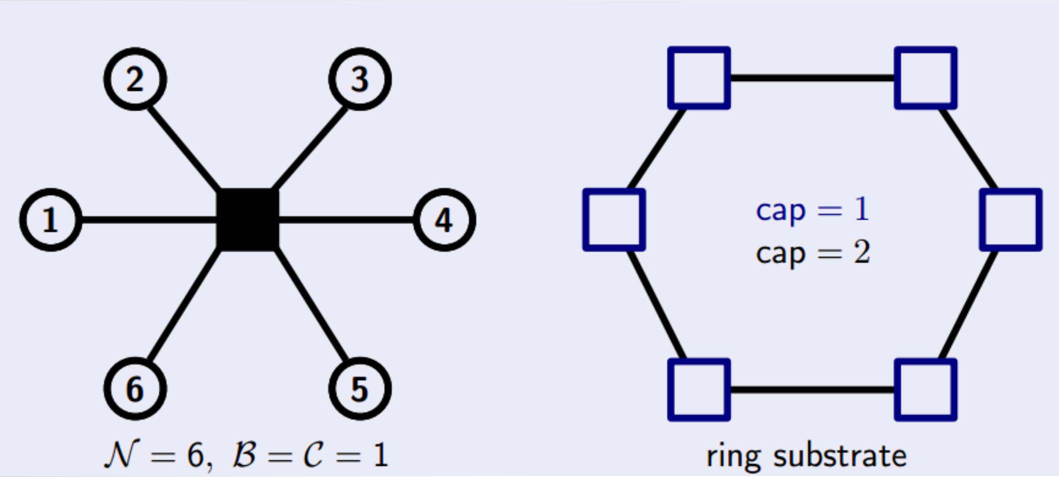
❑ Substrate: compute one unit, links two units

❑ VC Request



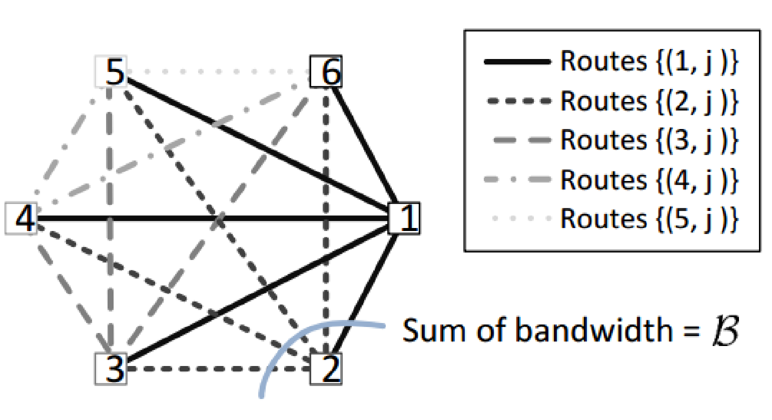
Thanks to Matthias Rost

On the Benefit of Hose (1)



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

- ❑ VC Request

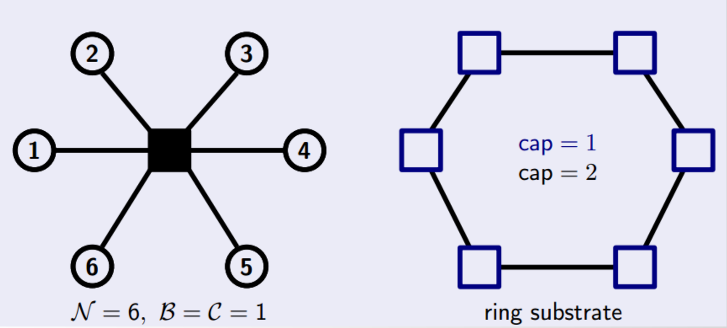


Impossible to map without splitting: need at least 5 independent paths to location where center is mapped!

Thanks to Matthias Rost

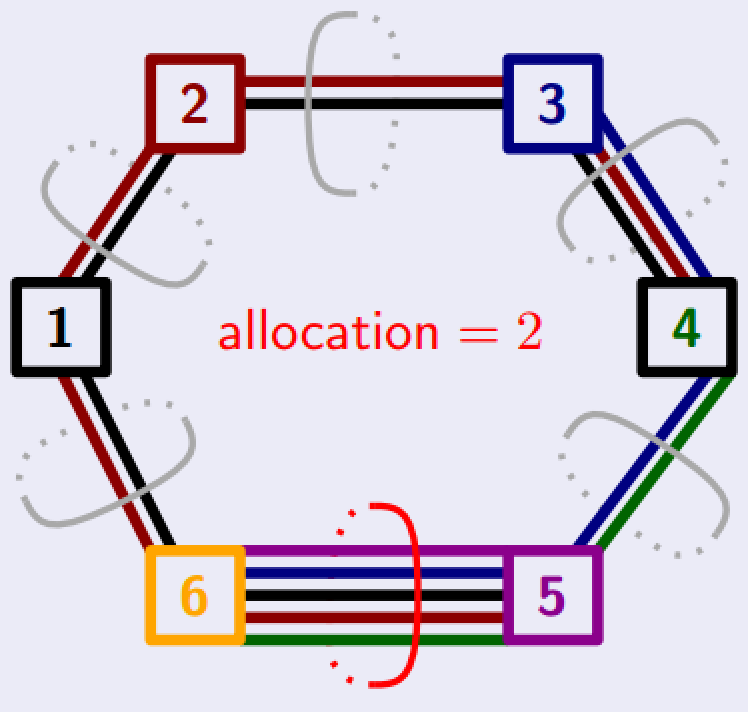
On the Benefit of Hose (2)

❑ In Hose model, it works!



$\mathcal{N} = 6,\ \mathcal{B} = \mathcal{C} = 1$

ring substrate



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

Own Literature (1)

General VNEP:

- 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.

Virtual Cluster:

- 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.

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

Own Literature (2)

Online Resource Allocation and Embeddings:

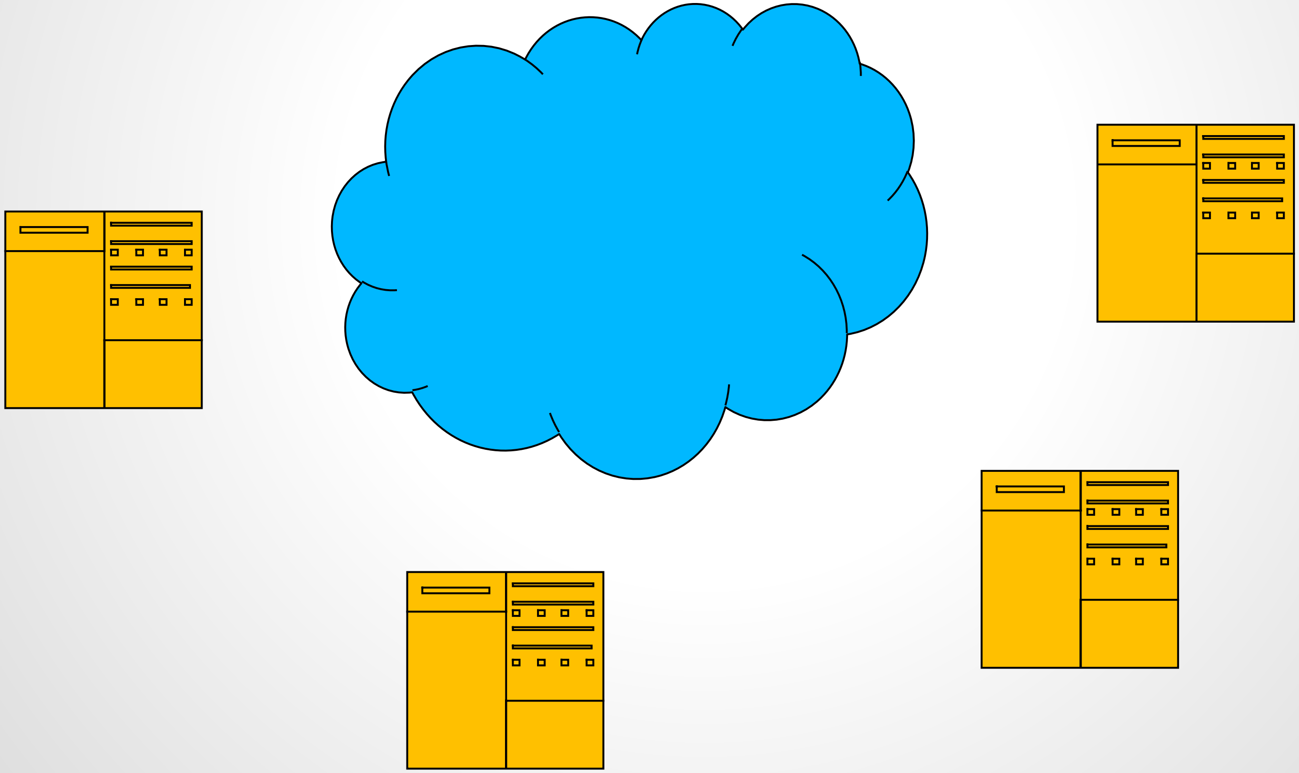
- Competitive Strategies for Online Cloud Resource Allocation with Discounts: The 2-Dimensional Parking Permit Problem
  Xinhui Hu, Arne Ludwig, Andrea Richa, and Stefan Schmid.
  35th IEEE International Conference on Distributed Computing Systems (**ICDCS**), Columbus, Ohio, USA, June 2015.

- The Wide-Area Virtual Service Migration Problem: A Competitive Analysis Approach
  Marcin Bienkowski, Anja Feldmann, Johannes Grassler, Gregor Schaffrath, and Stefan Schmid.
  IEEE/ACM Transactions on Networking (**ToN**), Volume 22, Issue 1, February 2014.

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

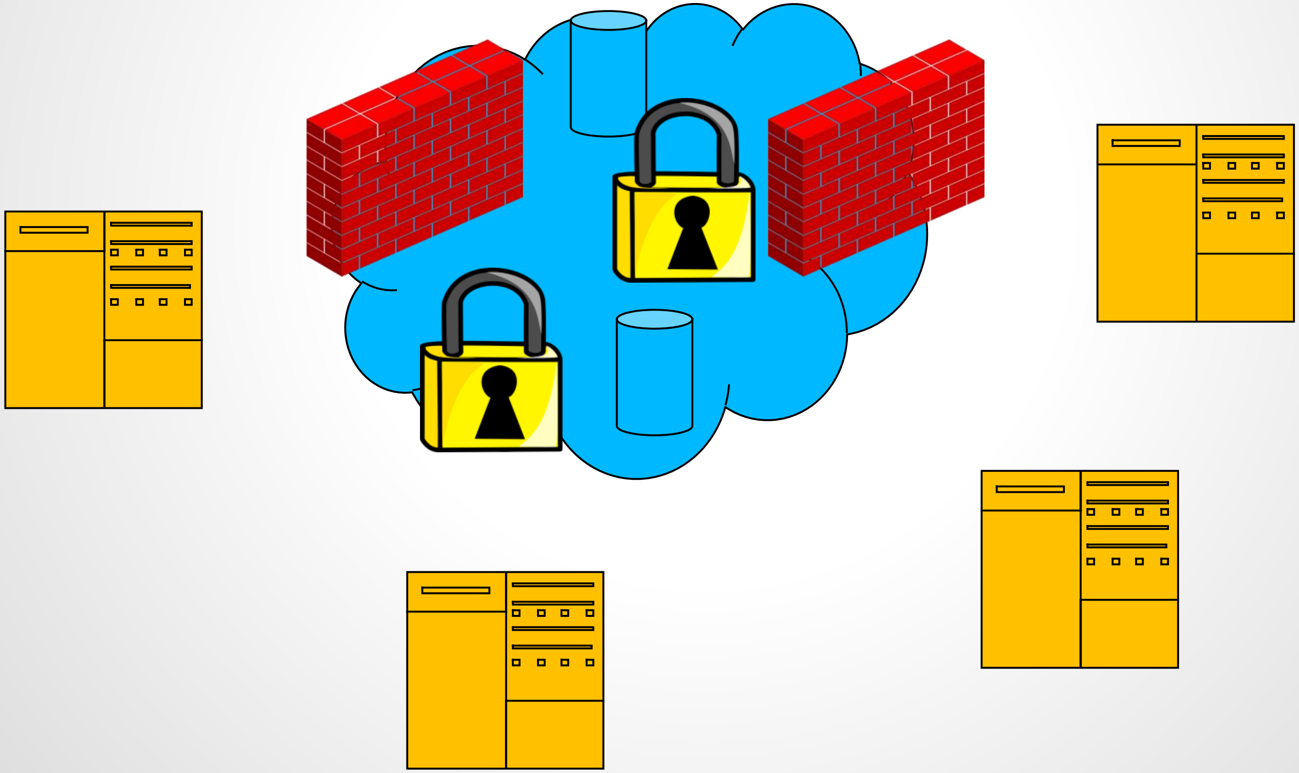
How to Exploit Flexibilities?
Example 2: Service Chain Embeddings

- The Internet?



How to Exploit Flexibilities?
Example 2: Service Chain Embeddings

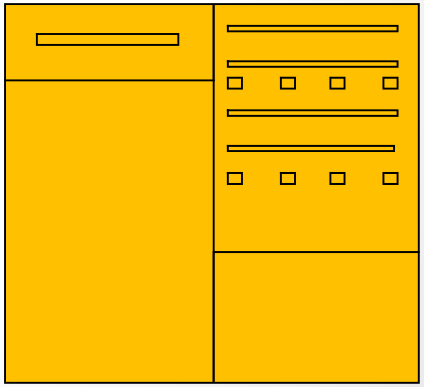
- The Internet today: # middleboxes ≈ # routers!



NFV = Flexible Allocation

- NFV: Virtualize the middlebox
  - SW middlebox in runs in VM…
  - … e.g., on a universal node

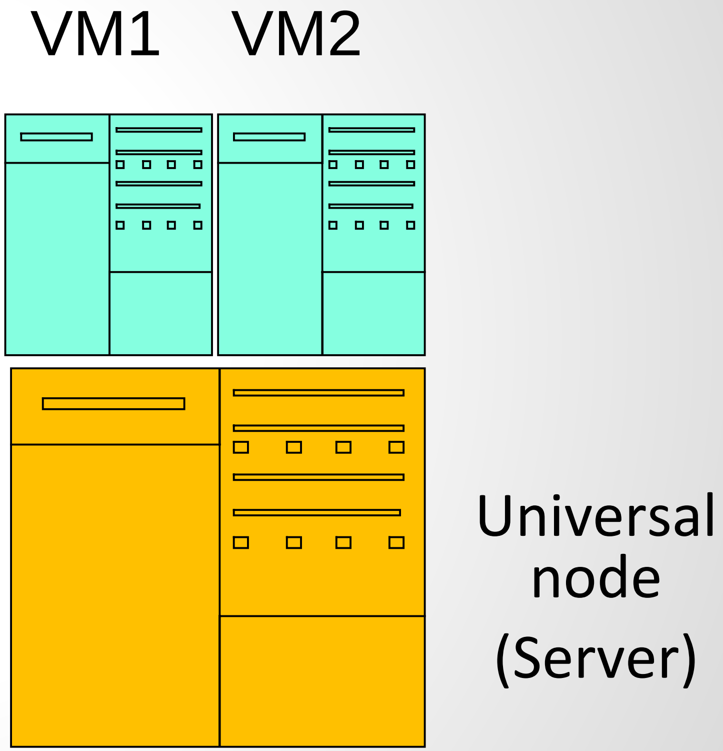
- Benefit:
  - Flexible and fast deployment
  - Can re-program it



Universal node
(Server)

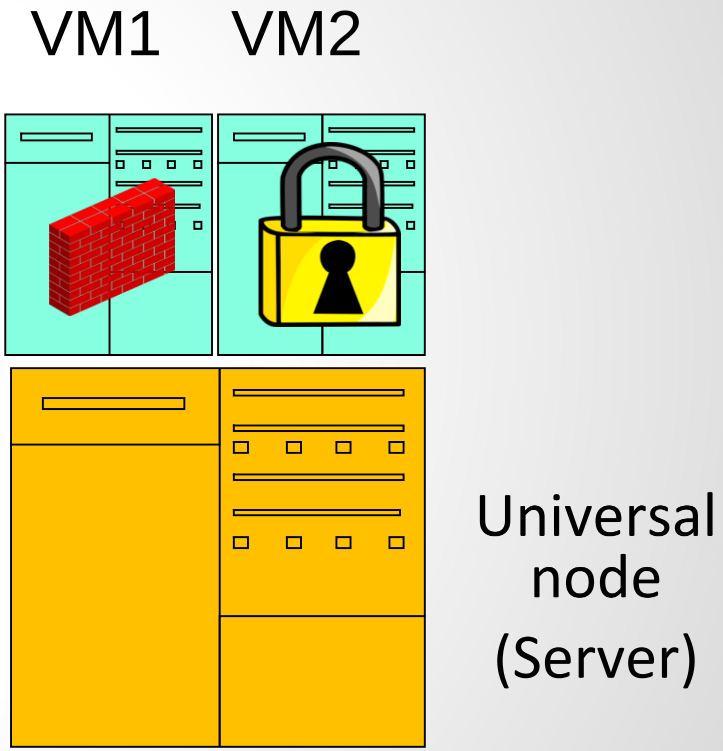
NFV = Flexible Allocation

- NFV: Virtualize the middlebox
  - SW middlebox in runs in VM…
  - … e.g., on a universal node
- Benefit:
  - Flexible and fast deployment
  - Can re-program it



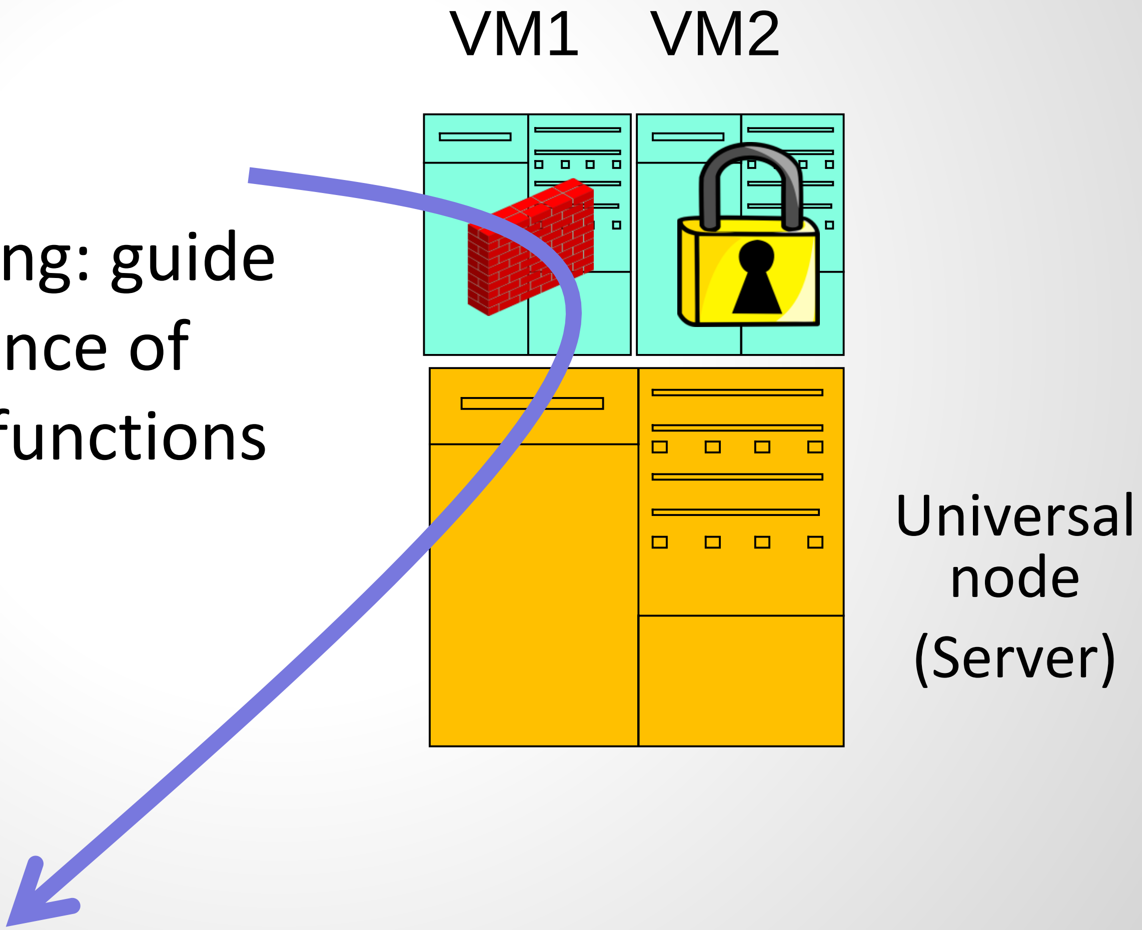
NFV = Flexible Allocation

- NFV: Virtualize the middlebox
  - SW middlebox in runs in VM…
  - … e.g., on a universal node
- Benefit:
  - Flexible and fast deployment
  - Can re-program it

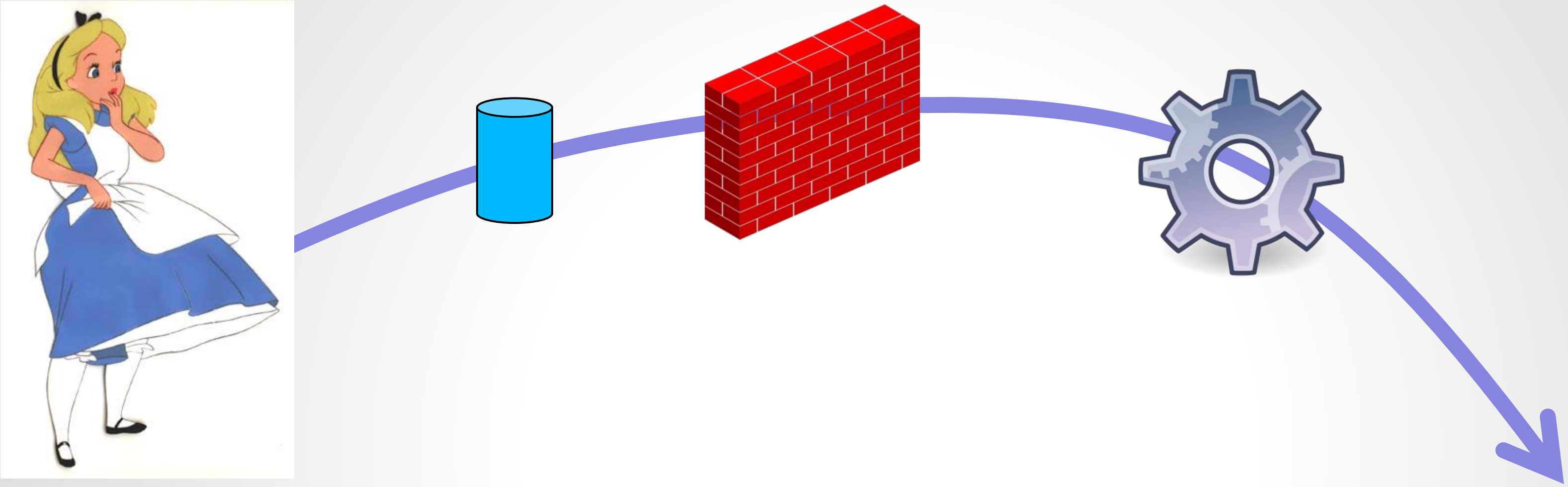


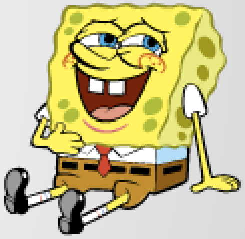
NFV = Flexible Allocation

Flexible traffic steering: guide flows through sequence of virtualized network functions



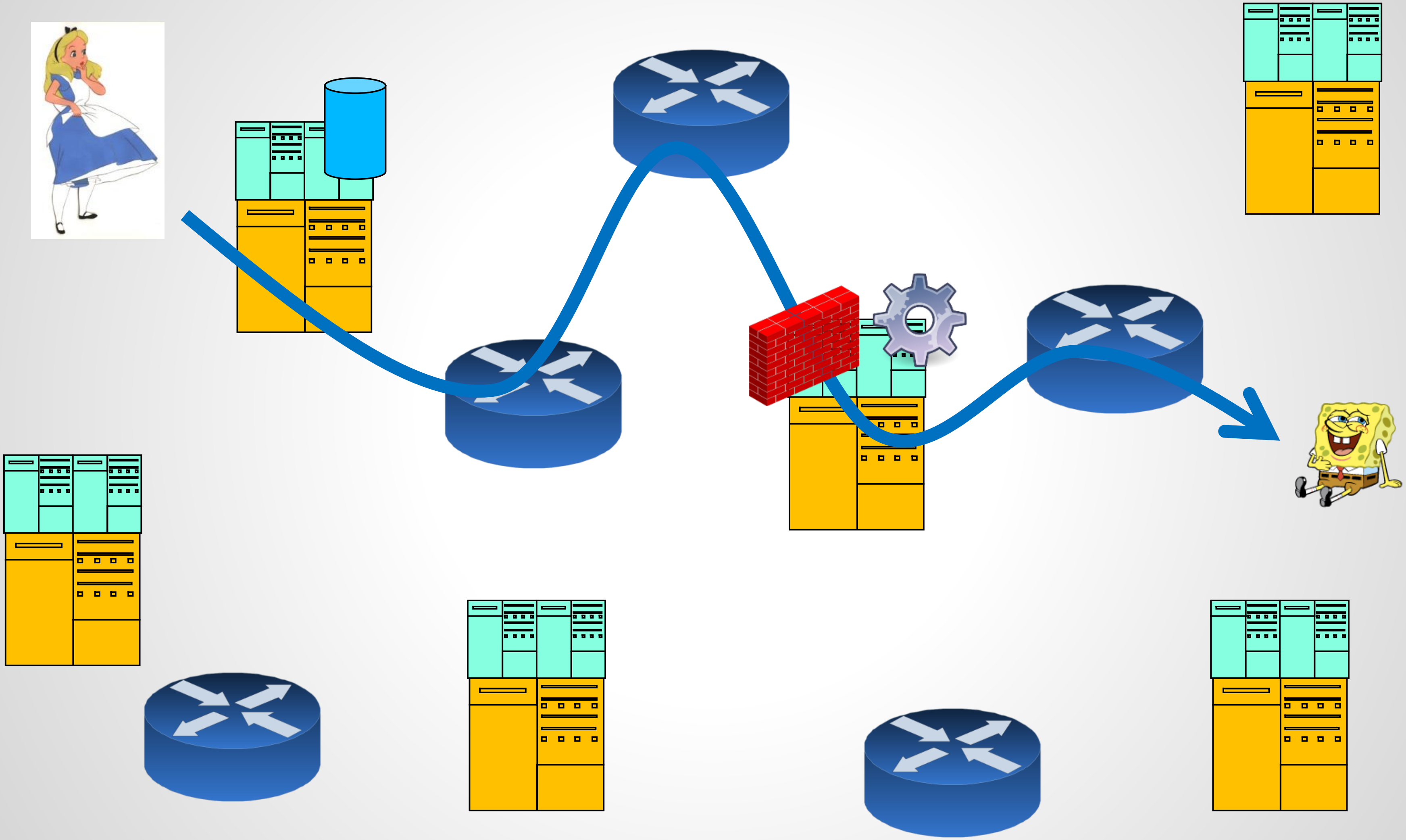
Service Chains



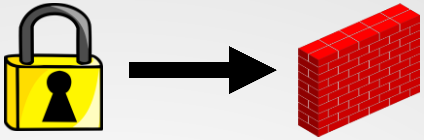


- Service chain = sequence of to be traversed network functions between A(lice) and B(ob)
- E.g., first go via proxy cache, then through firewall and then WAN optimizer

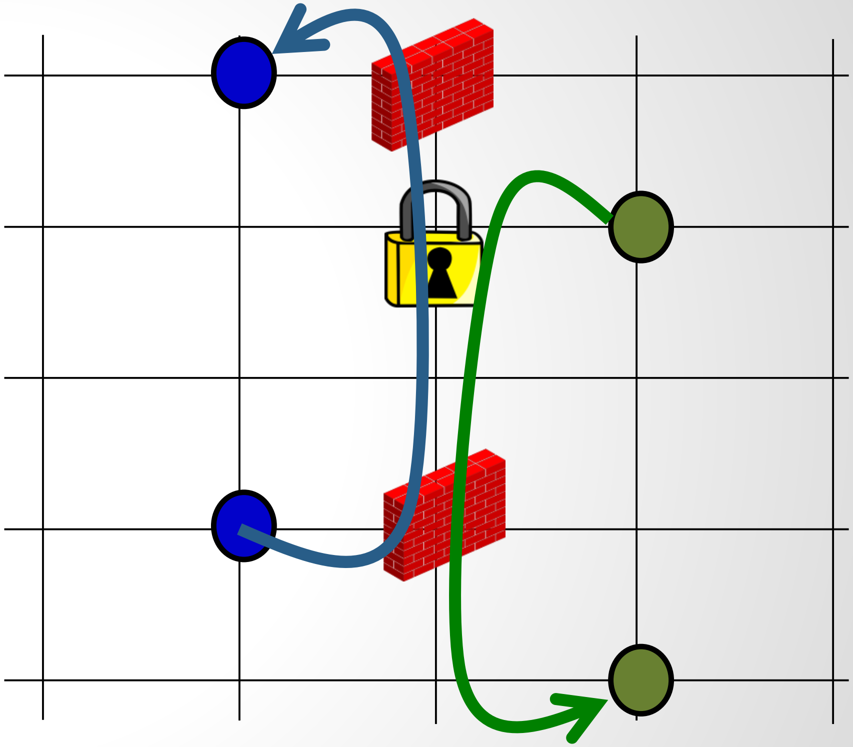
An Optimization Problem



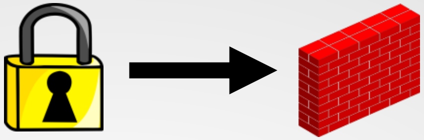
Model: Chain



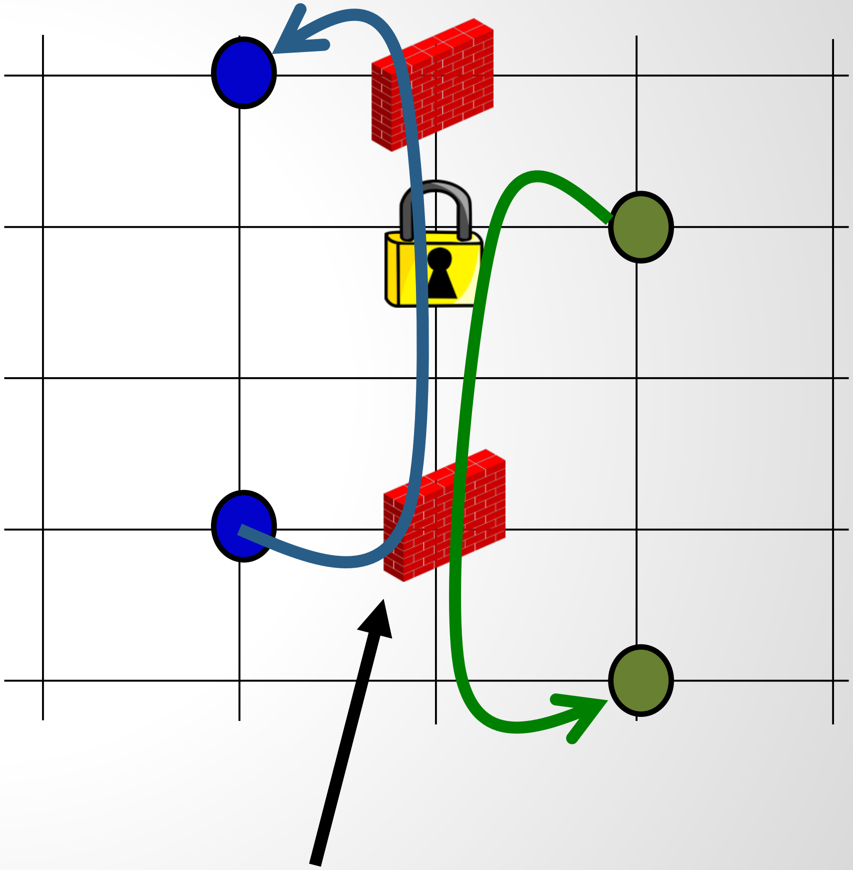
- n nodes
- L NF types: $F_1,\ldots, F_L$
- Instances of F_i: $f_i^{(1)}$, $f_i^{(2)}$,…
- A node can apply at most κ(v) functions
- Requests: $\sigma=(\sigma_1,\ldots,\sigma_k)$, $\sigma_i=(s_i,t_i)$
- For each σ_i, s_i and t_i need to be connected via a service chain $c_i=(f_1^{(x1)}, f_2^{(x2)},\ldots, f_L^{(xL)})$



Model: Chain

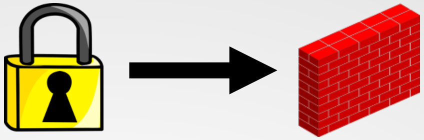


- n nodes
- L NF types: $F_1,\ldots, F_L$
- Instances of F_i: $f_i^{(1)}, f_i^{(2)},\ldots$
- A node can apply at most κ(v) functions
- Requests: $\sigma=(\sigma_1,\ldots,\sigma_k)$, $\sigma_i=(s_i,t_i)$
- For each σ_i, s_i and t_i need to be connected via a service chain $c_i=(f_1^{(x1)}, f_2^{(x2)},\ldots, f_L^{(xL)})$

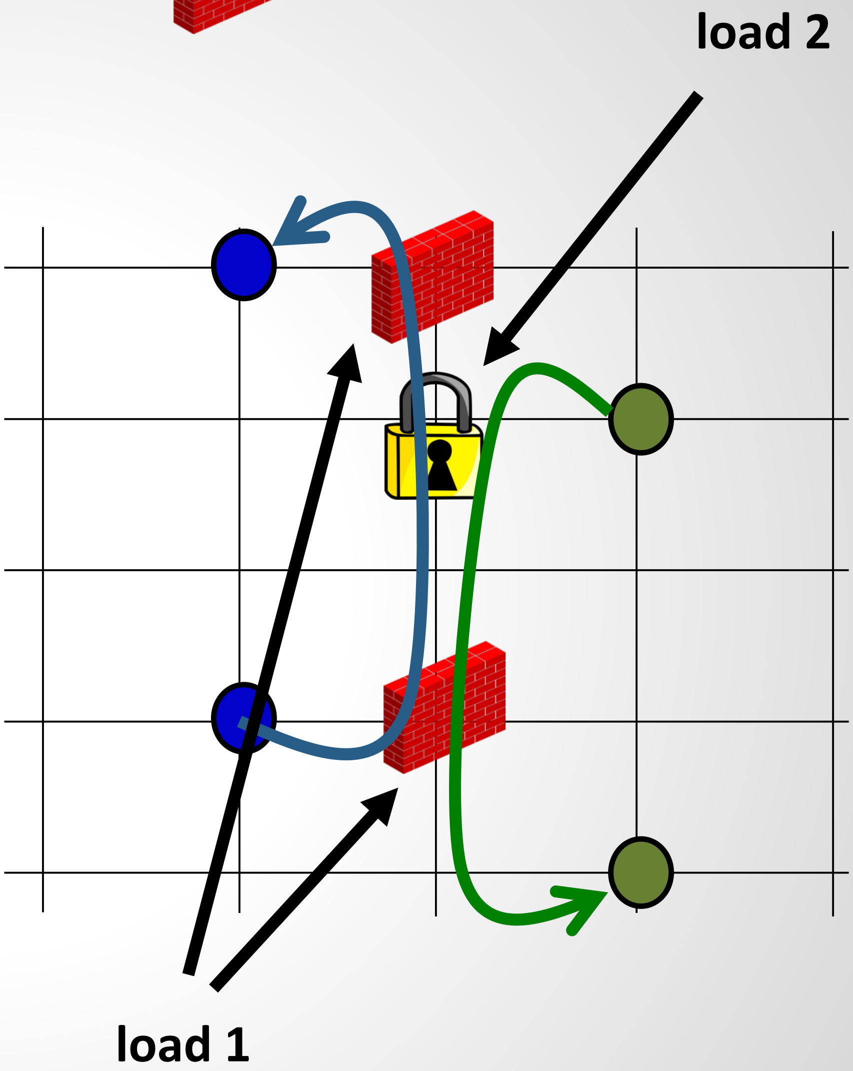


not applied to blue pair!

Model: Chain

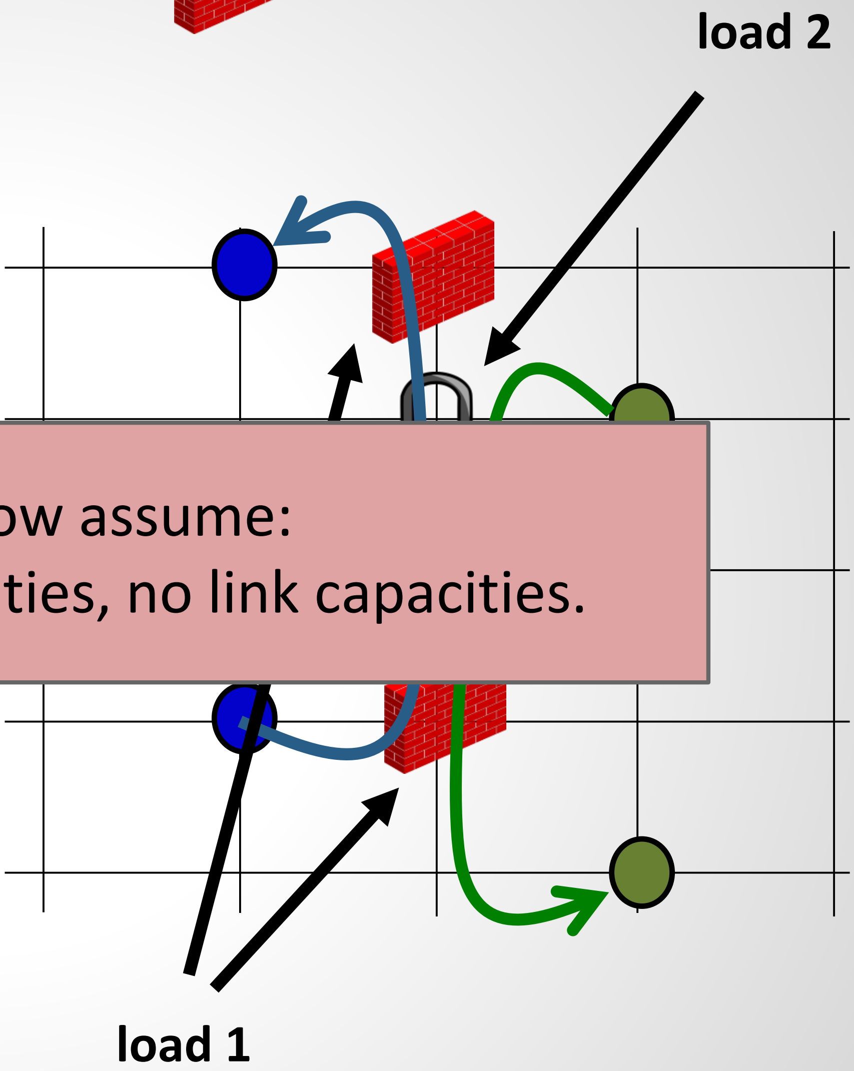


- n nodes
- L NF types: $F_1,\ldots, F_L$
- Instances of F_i: $f_i^{(1)}, f_i^{(2)},\ldots$
- A node can apply at most κ(v) functions
- Requests: $\sigma=(\sigma_1,\ldots,\sigma_k)$, $\sigma_i=(s_i,t_i)$
- For each σ_i, s_i and t_i need to be connected via a service chain $c_i=(f_1^{(x1)}, f_2^{(x2)},\ldots, f_L^{(xL)})$



Model: Chain

- n nodes
- L NF types: $F_1,\ldots, F_L$
- Instances of F_i: $f_i^{(1)}, f_i^{(2)},\ldots$
- A node car
  functions
- Requests: $\sigma=(\sigma_1,\ldots,\sigma_k)$, $\sigma_i=(s_i,t_i)$
- For each σ_i, s_i and t_i need to be connected via a service chain $c_i=(f_1^{(x1)}, f_2^{(x2)},\ldots, f_L^{(xL)})$



For now assume:
Only node capacities, no link capacities.

The SCEP Problem

- Maximum service chain embedding problem (SCEP)
- Given: sequence of requests: $\sigma=(\sigma_1,\ldots,\sigma_k)$, $\sigma_i=(s_i,t_i)$
- Constraints: (1) node capacity and (2) max path length r
- Goal: Admit and embed a **maximum number** of service chains without violating constraints

The SCEP Problem

- Maximum service chai

- Given: sequence of req

- Constraints: (1) node capacity and (2) max path length r

- Goal: Admit and embed a **maximum number** of service chains without violating constraints

Alternatively, we may support a bounded stretch!

Online Version of SCEP

- ❑ Requests arrive **one by one**
- ❑ On arrival of a request is to decide: **admit or reject**
- ❑ Admission: assign and embed the service chain
- ❑ Admitted requests cannot be canceled or rerouted
- ❑ *For now:* Service chains have **no duration**

What do we know?

Online SCEP:

- There exists an **O(log L)** competitive online algorithm
- **Ω(log L)** lower bound for any online algorithm

Offline SCEP:

- APX-hard for **unit capacities** and constant L ≥ 3
- Poly-APX-hard, when there is no bound on L
- Exact optimal solution via 0-1-ILP
- NP-completeness for constant L

What d

Good result in practice:
L is likely small!
(But capacities need to be at least log L.)

Online SCEP:

- There exists an **O(log L)** competitive online algorithm
- **Ω(log L)** lower bound for any online algorithm

Offline SCEP:

- APX-hard for **unit capacities** and constant L ≥ 3
- Poly-APX-hard, when there is no bound on L
- Exact optimal solution via 0-1-ILP
- NP-completeness for constant L

What do we know?

Online SCEP:

- There exists an **O(log L)** competitive online algorithm
- **Ω(log L)** lower bound for any online algorithm

Even holds if service chain is given!
(Like: path given in
online call admission)

Offline SCEP:

- APX-hard for **unit capa**
- Poly-APX-hard, when there is no bound on L
- Exact optimal solution via 0-1-ILP
- NP-completeness for constant L

What do we know?

Online SCEP:

- There exists an **O(log L)** competitive online algorithm
- **Ω(** algorithm

Reduction from
Maximum L-Set Packing

Offline SCEP:

- APX-hard for **unit capacities** and constant L ≥ 3
- Poly-APX-hard, when there is no bound on L
- Exact optimal solutio via 0-1 ILP
- NP-completeness for const

Reduction from
Maximum Independent Set

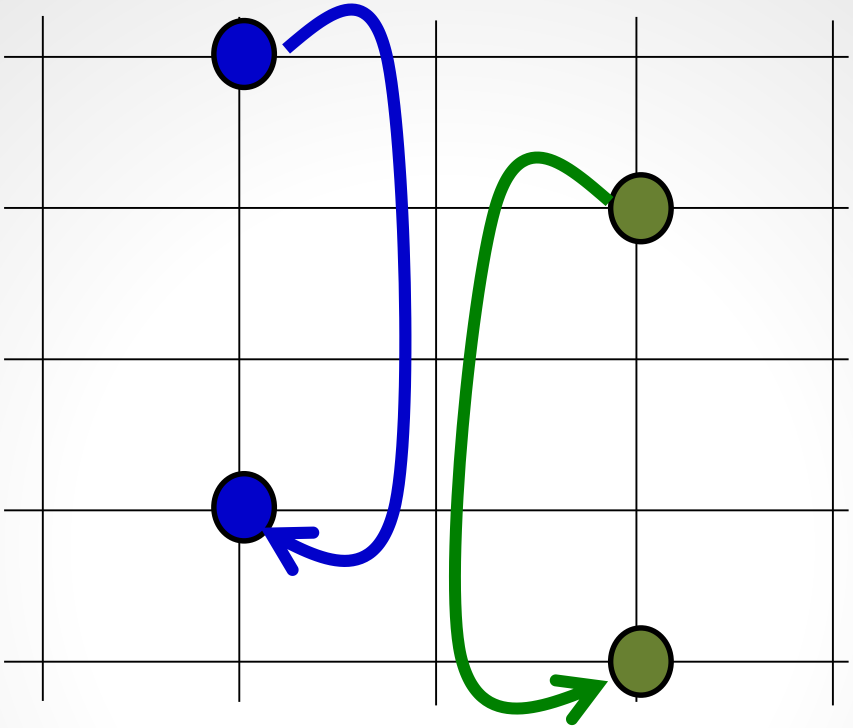
0-1 Program so in NP.

Preliminaries for Online Algorithm ACE

Ideas:

- **Preprocess**: Prune all chains which are too long
- If L is **small constant** (reasonable), can **generate all** possible chains *for a given request*: n^L
- Exploit connection to **online call admission**: accept only chains whose sum of node weights is small
- Node weight depends **exponentially** on current relative node load

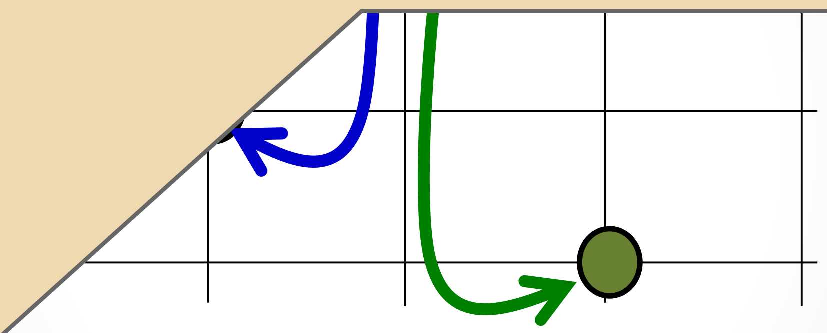
Background: Online Call Admission



- – Capacities on links (not nodes!)
- – Routing requests arrive online
- – Route (unsplittable!) is subject to optimization
- – Goal: Want to accept as many requests as possible

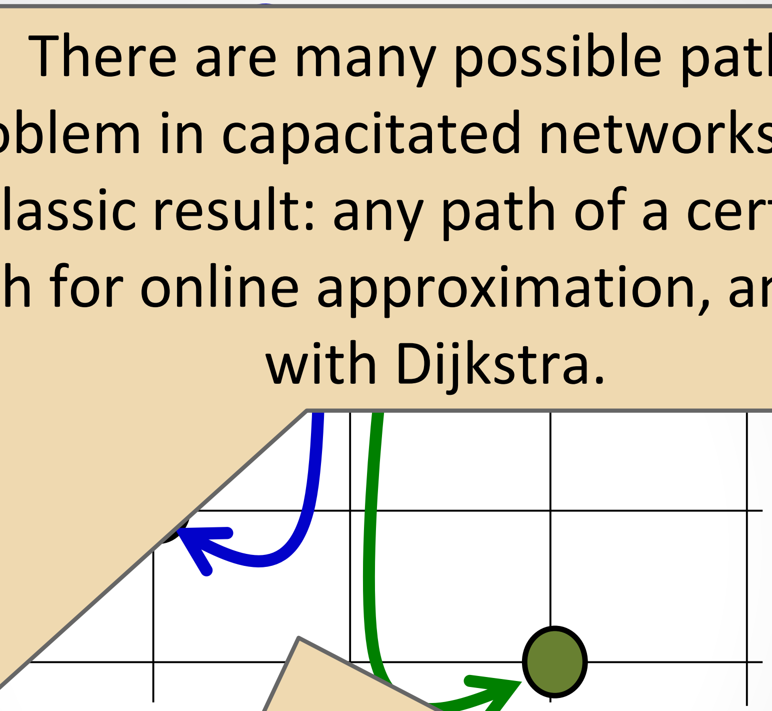
Background: Online Call Admission

There are many possible paths!
A hard problem in capacitated networks, even offline.
However, classic result: any path of a certain property is good enough for online approximation, and can be found with Dijkstra.



- Capaci on links (not nodes!)
- Routi g requests arrive online
- Route (unsplittable!) is subject to optimization
- Goal: Want to accept as many requests as possible

Background: Online Call Admission



There are many possible paths!
A hard problem in capacitated networks, even offline.
However, classic result: any path of a certain property is good enough for online approximation, and can be found with Dijkstra.

- Capaci on links
- Routing requests a
- Route (unsplittable
- Goal: Want to accept as many requests as possible

In our case, we will focus on node capacities, not links. No routing needed, can generate all chains.

Preliminaries for Online Algorithm ACE

ACE = Admission Control and **C**hain **E**mbedding Algorithm

Idea: Exploit connection to Virtual Circuit routing! Let‘s define a cost for hosting a NF for a chain which is *exponential* in the *relative load* of the node

- relative load at node v before the j-th request:

$$\lambda_v(j) = \frac{\#\text{ admitted chains through } v}{\kappa(v)}$$

- cost of v before processing the j-th request:

$$w_v(j) = \kappa(v)(\mu^{\lambda_v(j)} - 1),$$

where *m* = 2L + 2

Preliminaries for Online Algorithm ACE

ACE = Admission Control and **C**hain **E**mbedding Algorithm

Idea: Exploit connection to Virtual Circuit routing! Let‘s define a cost for hosting a NF for a chain which is *exponential* in the *relative load* of the

We will respect capacity constraints: ensure that the relative load never exceeds 1

- relative load at node ore the j-th request:

$$\lambda_v(j) = \frac{\#\ \text{admitted chains through } v}{\kappa(v)}$$

- cost of v before processing the j-th request:

$$w_v(j) = \kappa(v)(\mu^{\lambda_v(j)} - 1),$$

where *m* = 2L + 2

Preliminaries for Online Algorithm ACE

ACE = Admission Control and **C**hain **E**mbedding Algorithm

Idea: Exploit connection to Virtual Circuit routing! Let‘s define a cost for hosting a NF for a chain which is *exponential* in the *relative load* of the node

- relative load at node v before the j-th req

$$\lambda_v(j) = \frac{\text{\# admitted chains through } v}{\kappa(v)}$$

We need to assume that this is at least log L

- cost of v before processing the j-th request:

$$w_v(j) = \kappa(v)(\mu^{\lambda_v(j)} - 1),$$

where *m* = 2L + 2

Preliminaries for Online Algorithm ACE

ACE = Admission Control and **C**hain **E**mbedding Algorithm

Idea: Exploit connection to Virtual Circuit routing! Let's define a cost for hosting a NF for a chain which is *exponential* in the *relative load* of the node

- relative load at node v before the j-th request:

$$\lambda_v(j) = \frac{\#\ \text{admitte}}{\kappa(}$$

The cost is exponential in the relative load.

- cost of v before processing the j-th request:

$$w_v(j) = \kappa(v)(\mu^{\lambda_v(j)} - 1),$$

where *m* = 2L + 2

Online Algorithm: ACE

Algorithm ACE is very simple:

- When request σ_j arrives, check if there exists a chain c_j , s.t.

  1. σ_j can be routed along c_j on a path of valid length r

  2. $$\sum_{v \in c_j} \frac{w_v(j)}{\kappa(v)} \leq L$$

- If such a chain c_j exists, then admit σ_j and assign it to c_j. Otherwise, reject σ_j.

Analysis of ACE

Theorem: Assume, $\min_v(k(v)) \geq \log m$. Then ACE never violates capacity and length constraints and is O(log L) competitive.

Proof sketch:

- Lemma 1: Requests admitted by ACE are feasible and respect capacity constraints.
- Lemma 2: Sum of node costs (over all nodes) after last request k is proportional (up to L log m factors) to the number of accepted requests |A|

$$(2\mathrm{L} \log \mu)|A| \geq \sum_v w_v(k+1)$$

- Lemma 3: Let A* be the set of requests accepted by OPT but not ACE. Then:

$$|A^*| \cdot \mathrm{L} \leq \sum_v w_v(k+1)$$

Analysis of ACE

Theorem: Assume, $\min_v(k(v)) \ge \log m$. Then ACE never violates capacity and lengt L) competitive.

By contradiction of how ACE accepts requests.

Proof sketch:

- Lemma 1: Requests admitted by ACE are feasible and respect capacity constraints.
- Lemma 2: Sum of node costs (over all nodes) after last request k is proportional (up to L log m factors) to the number of accepted requests |A|

$$(2\mathsf{L}\log\mu)|A| \ge \sum_v w_v(k+1)$$

- Lemma 3: Let A* be the set of requests accepted by OPT but not ACE. Then:

$$|A^*|\cdot \mathsf{L} \le \sum_v w_v(k+1)$$

Analysis of ACE

Theorem: Assume, $\min_v(k(v)) \geq \log m$. Then ACE never violates capacity and length constraints and is O(log L) competitive.

Proof sketch:

- Lemma 1: Requests admitted by ACE are feasible and respect capacity constraints.
- Lemma 2: Sum of node costs (ov [illegible] k is proportional (up to L log m factors) t [illegible] number of accepted requests |A|

By induction over accepted requests.

$$(2L\log\mu)|A| \geq \sum_v w_v(k+1)$$

- Lemma 3: Let A* be the set of requests accepted by OPT but not ACE. Then:

$$|A^*|\cdot L \leq \sum_v w_v(k+1)$$

Analysis of ACE

Theorem: Assume, $\min_v(k(v)) \geq \log m$. Then ACE never violates capacity and length constraints and is O(log L) competitive.

Proof sketch:

- Lemma 1: Requests admitted by ACE are feasible and respect capacity constraints.
- Lemma 2: Sum of node costs (o proportional (up to L log m fac

$$(2\mathsf{L}\log\mu)$$

By the fact that costs increase monotonically and also OPT needs to respect capacities.

- Lemma 3: Let A* be the set of reques ccepted by OPT but not ACE. Then:

$$|A^*| \cdot \mathsf{L} \leq \sum_v w_v(k+1)$$

Analysis of ACE

The theorem then follows: By definition

$$|A_{\mathrm{OFF}}| \leq |A| + |A^*|$$

By Lemma 3:

$$|A_{\mathrm{OFF}}| \leq |A| + \frac{1}{\mathsf{L}} \sum_v w_v(k+1)$$

By Lemma 2:

$$|A_{\mathrm{OFF}}| \ \leq |A| + 2 \cdot (\log \mu) \cdot |A| = (1 + 2 \log \mu)|A|$$

Theorem: violates

capacity a tive.

Proof sket

- Lemma 1: pacity constraint
- Lemma 2: is proportion requests |A|

$$(2\mathsf{L} \log \mu)|A| \geq \sum_v w_v(k+1)$$

- Lemma 3: Let A* be the set of requests accepted by OPT but not ACE. Then:

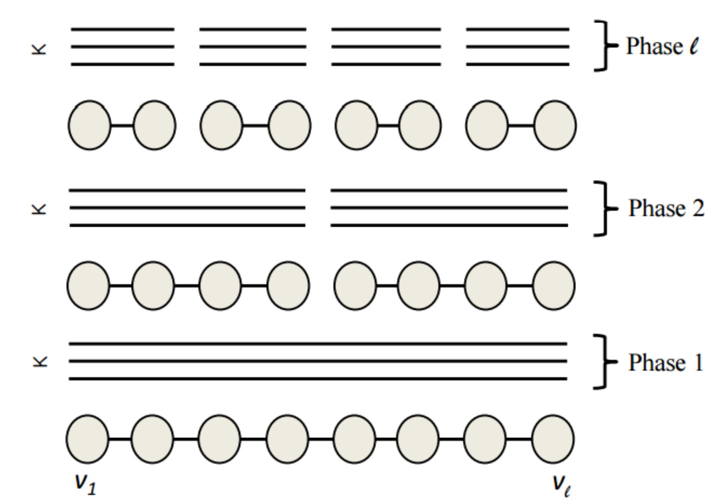
$$|A^*| \cdot \mathsf{L} \leq \sum_v w_v(k+1)$$

Lower Bound

Theorem: Assume, k ≥ log m. Any online algorithm for SCEP must have a competitive ratio of at least Ω(log L).

Proof:

- Requests come in log L phases
- In phase i: 2^i groups of k requests sharing subsets of size L/2^i.
- Tradeoff: accepting early means missing many future requests!
- Adversary stops when online algorithm admitted at most 2^j+1*k / log L requests till phase j. (j must exist)
- OPT rejects all requests except for 2^j*k in phase j.



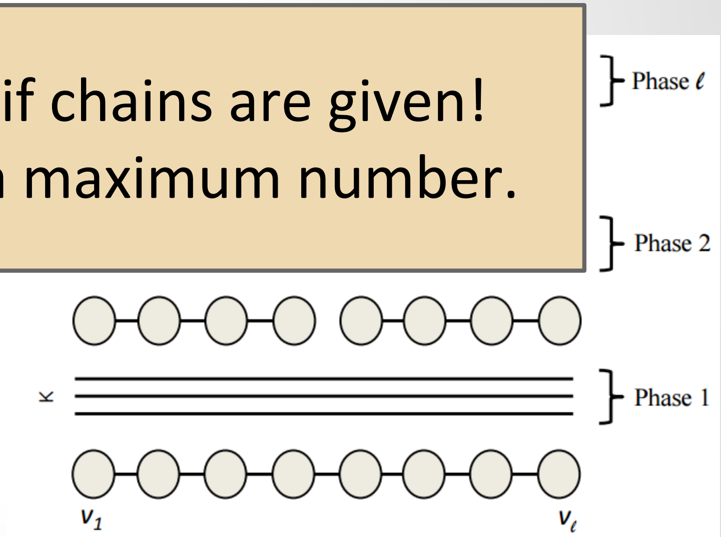
Lower Bound

Theorem: Assume, k ≥ log m. Any online algorithm for SCEP must have a competitive ratio of at least Ω(log L).

Proof:

- Requests come in log L phases
- In pha
  sharin
- Trade
  missing many future requests!
- Adversary stops when online algorithm admitted at most 2^j+1*k / log L requests till phase j. (j must exist)
- OPT rejects all requests except for 2^j*k in phase j.

Lower bound even holds if chains are given!
And goal is just to accept a maximum number.



Offline SCEP

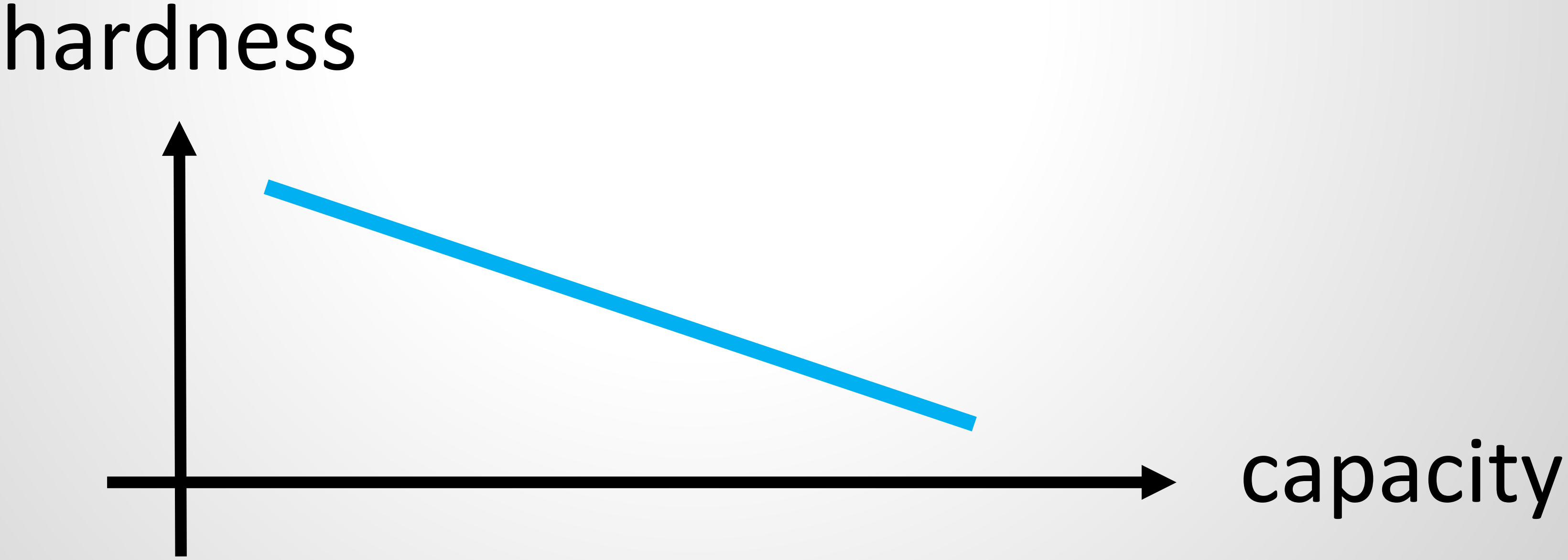
Theorem: Let $L \geq 3$ be a constant and $\kappa(v) = 1$, for all v. Then the offline SCEP is APX-hard.

Proof idea:

- Reduction of Maximum L-Set Packing Problem (LSP) to SCEP
- Approximation preserving reduction
- LSP is APX-complete

Offline SCEP: Inapproximability Result

Theorem: Let $L \geq 3$ be a constant and $\kappa(v) = 1$, for all v. Then the offline SCEP is APX-hard and not approximable within L^{ε} for some $\varepsilon > 0$. Without a bound on the chain length the SCEP with $\kappa(v) = 1$, for all nodes v, is Poly-APX-hard.



Offline SCEP: Inapproximability Result

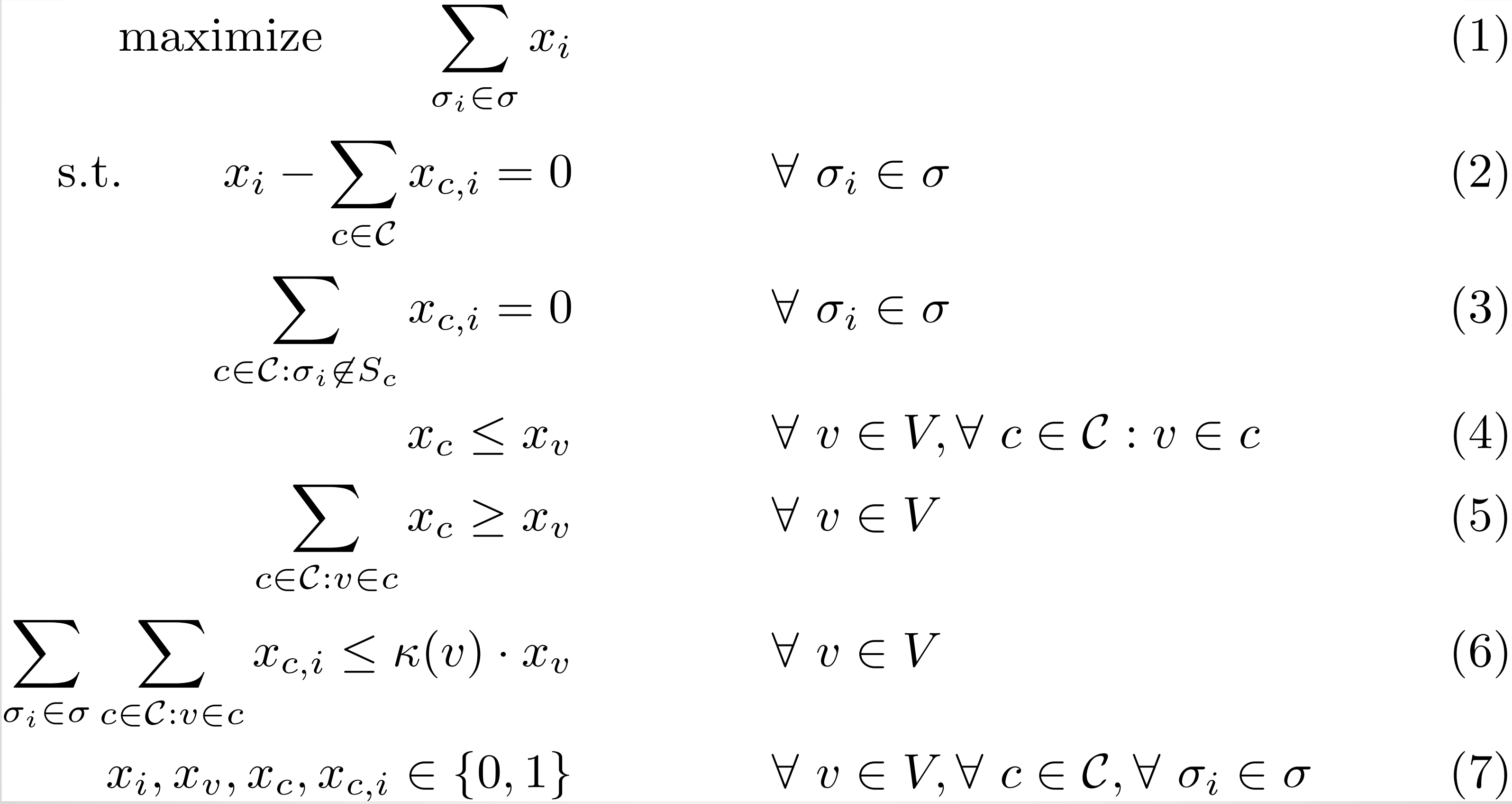
Theorem: Let $L \geq 3$ be a constant and $\kappa(v) = 1$, for all v. Then the offline SCEP is APX-hard and not approximable within L^{ε} for some $\varepsilon > 0$. Without a bound on the chain length the SCEP with $\kappa(v) = 1$, for all nodes v, is Poly-APX-hard.

Proof idea:

- Reduction of Maximum Independent Set Problem (MIS) to SCEP
- Approximation preserving reduction
- MIS is APX-complete and cannot be approximated within L^{ε} for some $\varepsilon > 0$.
- For graphs without degree bound, the MIS is Poly-APX-complete.

0-1 Linear Program – NP-completeness

Exact optimal solution via 0-1-ILP



$$
\begin{aligned}
&\text{maximize} && \sum_{\sigma_i \in \sigma} x_i && && (1)\\
&\text{s.t.} && x_i - \sum_{c \in \mathcal{C}} x_{c,i} = 0 && \forall\ \sigma_i \in \sigma && (2)\\
& && \sum_{c \in \mathcal{C} : \sigma_i \notin S_c} x_{c,i} = 0 && \forall\ \sigma_i \in \sigma && (3)\\
& && x_c \leq x_v && \forall\ v \in V, \forall\ c \in \mathcal{C} : v \in c && (4)\\
& && \sum_{c \in \mathcal{C} : v \in c} x_c \geq x_v && \forall\ v \in V && (5)\\
& && \sum_{\sigma_i \in \sigma} \sum_{c \in \mathcal{C} : v \in c} x_{c,i} \leq \kappa(v) \cdot x_v && \forall\ v \in V && (6)\\
& && x_i, x_v, x_c, x_{c,i} \in \{0, 1\} && \forall\ v \in V, \forall\ c \in \mathcal{C}, \forall\ \sigma_i \in \sigma && (7)
\end{aligned}
$$

Summary

- Network virtualization introduces algorithmic flexibilities

- Don‘t be afraid, even if others say it is hard! ☺

- A first look at provably good online admission control and embedding of service chains

Own Literature

- 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.

- Network Service Chaining with Optimized Network Function Embedding Supporting Service Decompositions
  Sahel Sahhaf, Wouter Tavernier, Matthias Rost, Stefan Schmid, Didier Colle, Mario Pickavet, and Piet Demeester.
  Journal Computer Networks (**COMNET**), Elsevier, to appear.

Thank you!