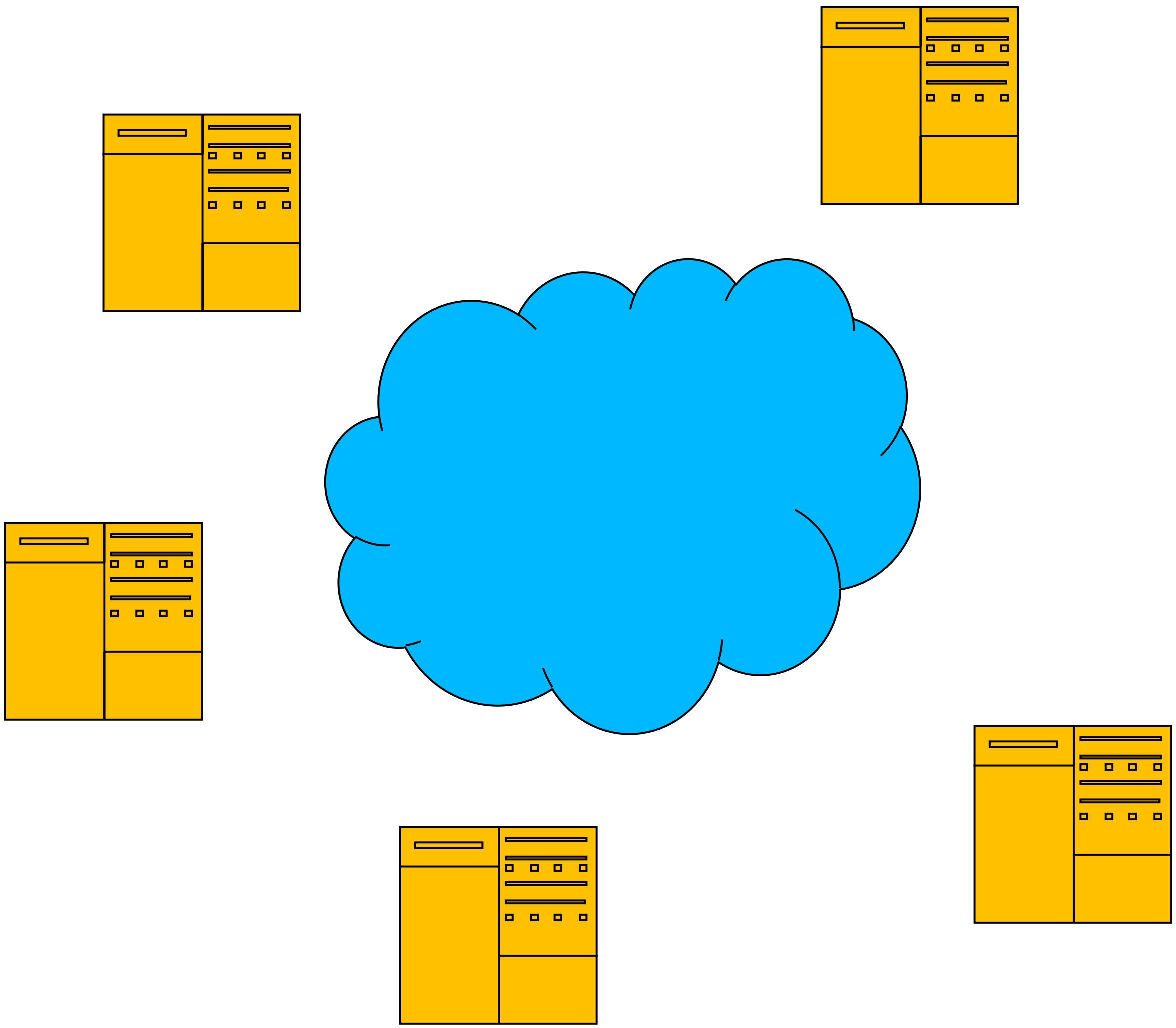
Online Admission Control and Embedding of Service Chains

Tamás Lukovszki[1] and Stefan Schmid[2]

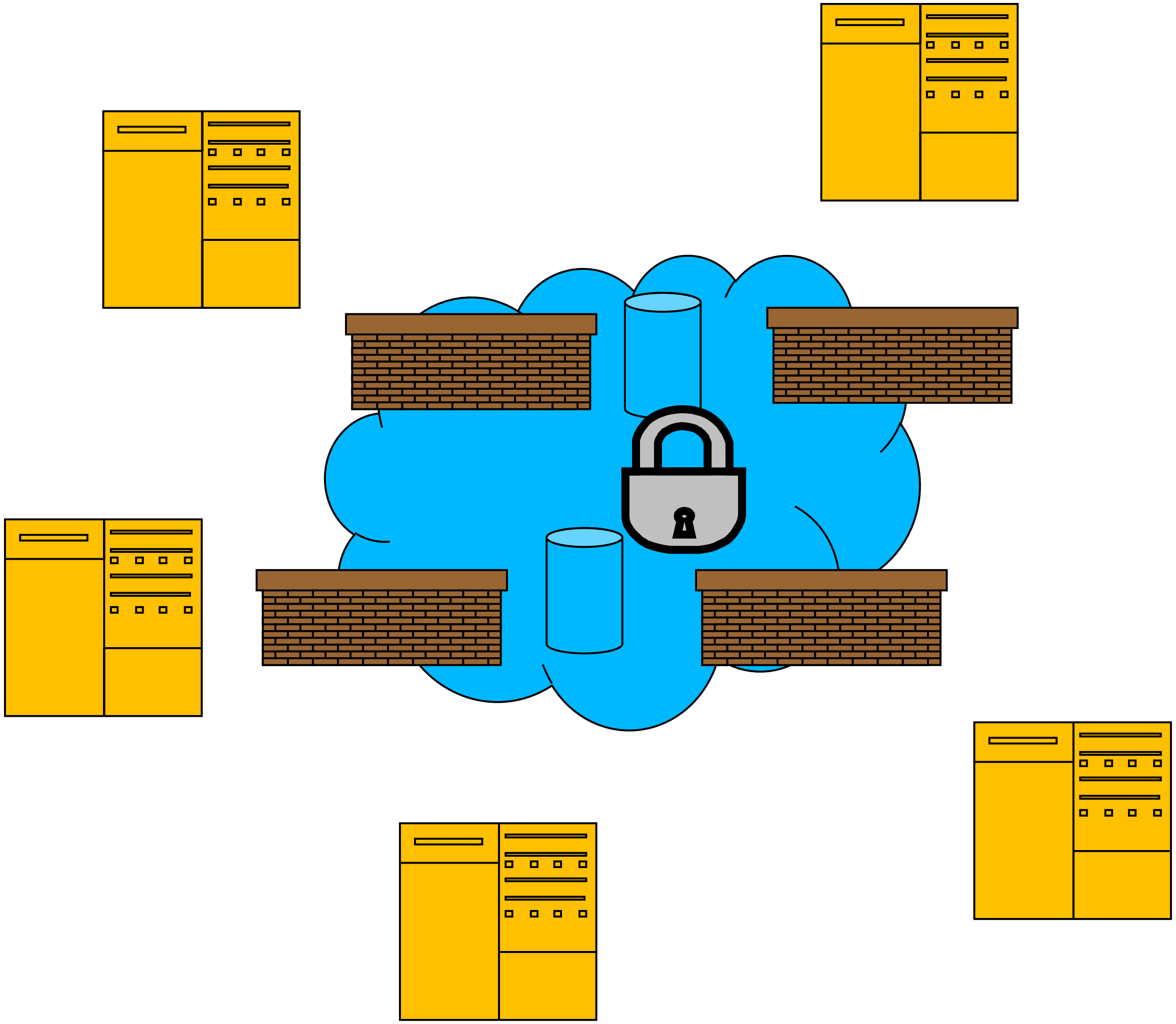
1 Eötvös Loránd University, Budapest, Hungary
lukovszki@inf.elte.hu

2 TU Berlin & Telekom Innovation Laboratories, Berlin, Germany
stefan.schmid@tu-berlin.de

The Internet?



The Internet!



Middleboxes

The Internet contains many middleboxes

- Middlebox: aka “a bump in the wire”
- Firewalls, NATs, proxies, caches, WAN optimizer, encryption…
- Studies show: number of middleboxes in the order of the number of routers!

Middleboxes

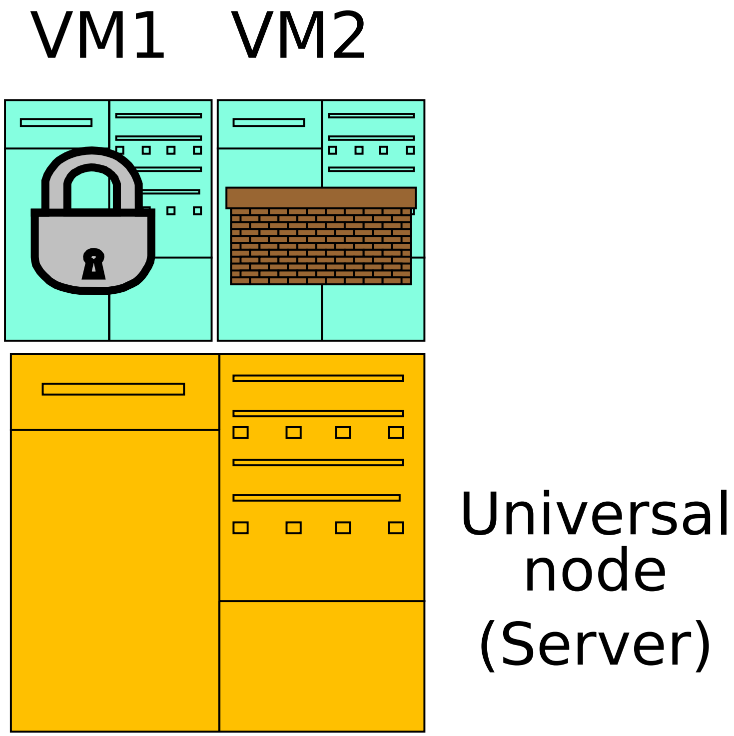
The Internet contains many middleboxes

- Middlebox: aka “a bump in the wire”
- Firewalls, NATs, proxies, caches, WAN optimizer, encryption…
- Studies show: number of middleboxes in the order of the number of routers!

Problem: Middleboxes are expensive, cumbersome to deploy and manage…

Trend: NFV = Flexible Allocation

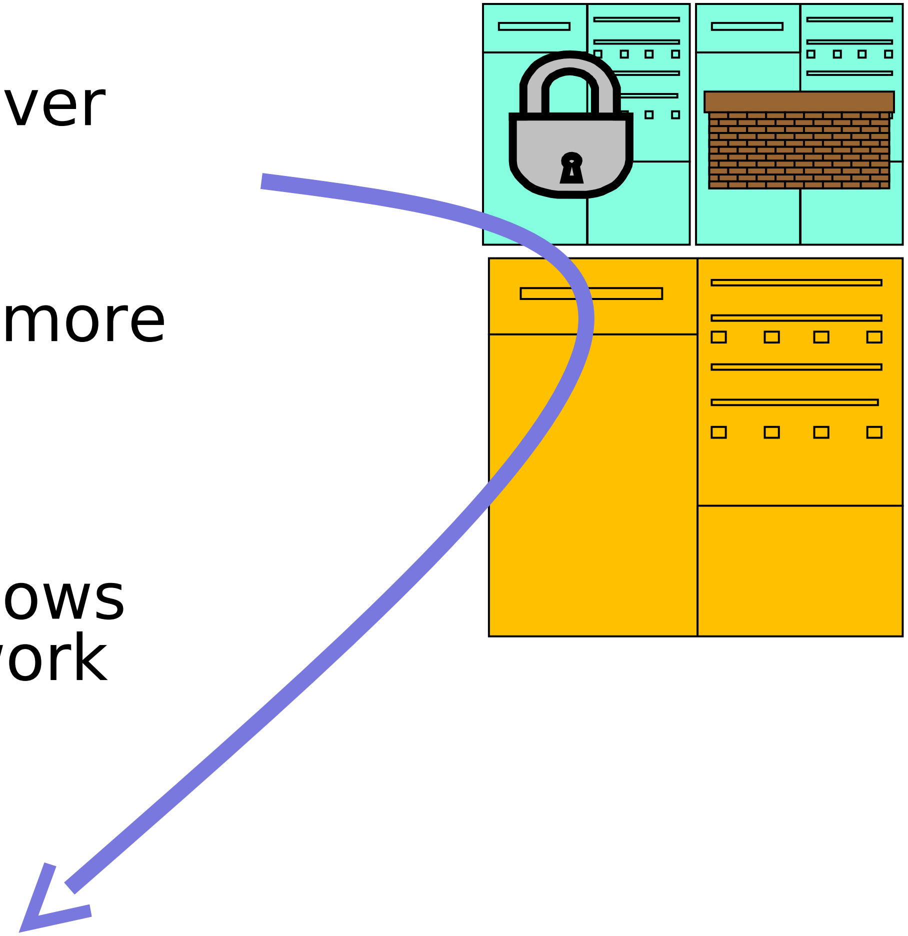
- Trend: Network Function Virtualization (NFV)
- Virtualize the middlebox:
  - Running in software, e.g., running in a VM
  - Many middlebox templates run on a “universal node”
- Benefit:
  - Flexible and fast deployment!
  - Can even program / reprogram it



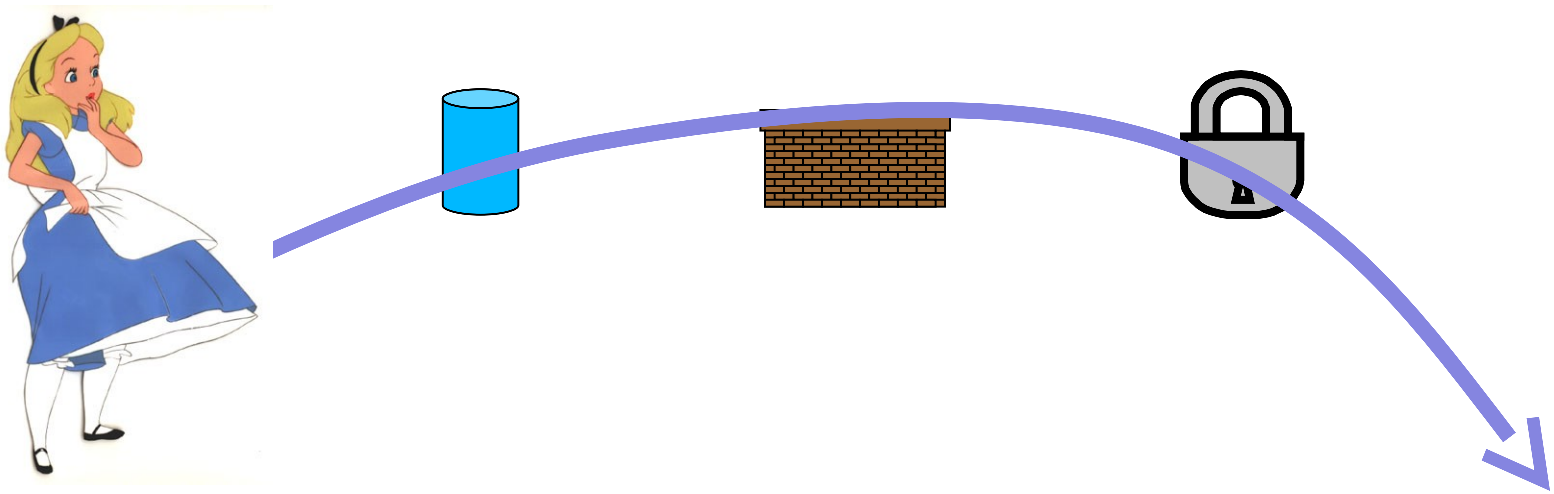
NFV + SDN

Software-Defined Networking

- Outsources control over switches to software
- Renders networking more flexible
- For example traffic engineering: guide flows through Virtual Network Functions



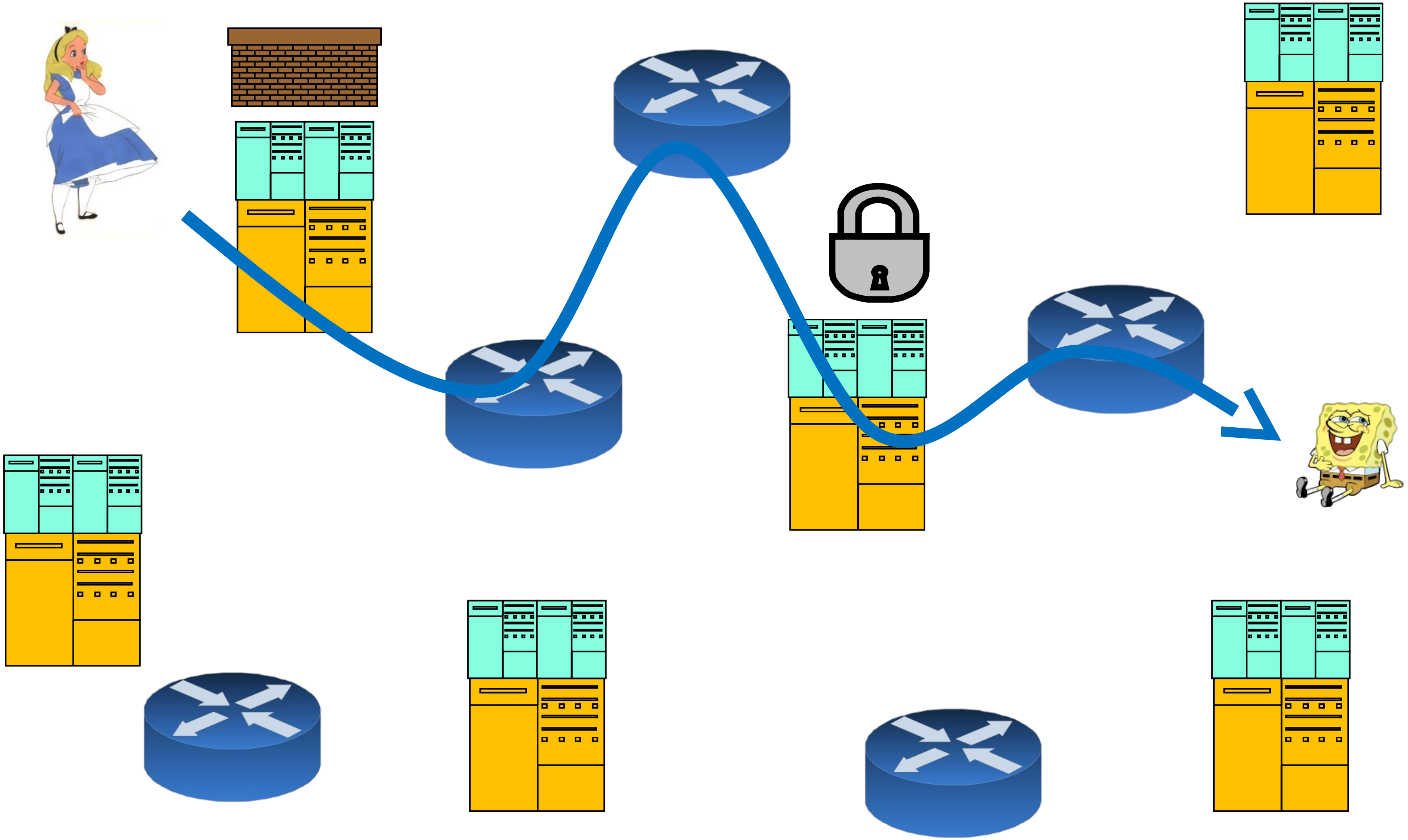
Service Chains



– Service chain = sequence of to be traversed network functions between A and B

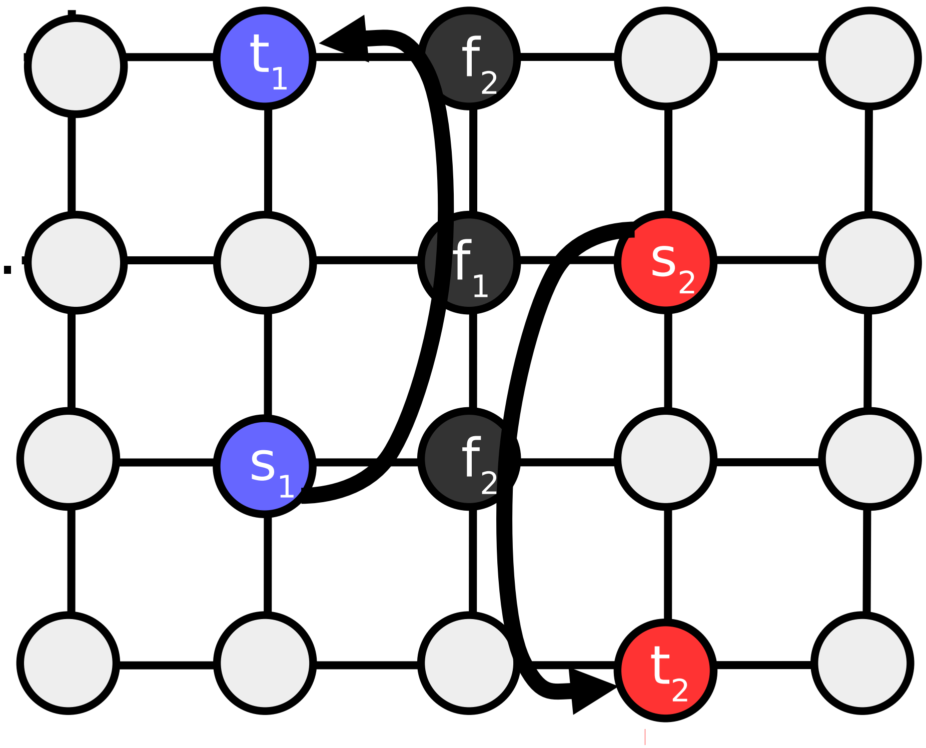
– E.g., first go via proxy cache, then through NAT and then WAN optimizer

Our Problem



Model

- Network of n nodes
- L NF types: $F_1, \ldots, F_L$
- Instances of F_i: $f_i^{(1)}, f_i^{(2)}, \ldots$
- One node can host also more than 1 NFs
- Requests: $\sigma = (\sigma_1, \ldots, \sigma_k)$, $\sigma_i = (s_i, t_i)$
- For each σ_i, s_i and t_i neeeds to be connected via a service chain $c_i = (f_1^{(x1)}, f_2^{(x2)}, \ldots, f_L^{(xL)})$



Problem

Maximum service chain embedding problem (SCEP)

Given:

- Network $G=(V,E)$, $|V|=n$
- NFs
- Requests: $\sigma=(\sigma_1,\ldots,\sigma_k)$, $\sigma_i=(s_i,t_i)$

Constraints:

- $\kappa(v)$ is the maximum number of requests, for which node v in V can apply a NF
- path length (# hopps) for each chain must be at most R

Goal:

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

Results

On-line SCEP:

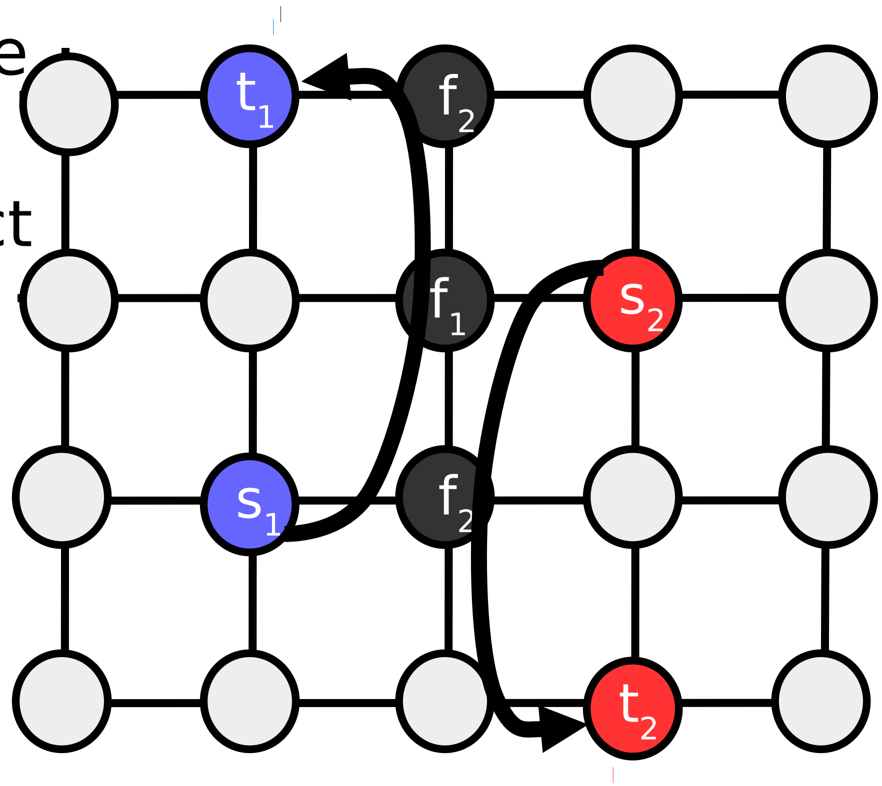
- **O(log L)**-competetive on-line algorithm
- **Ω(log L)** lower bound on the competetive ratio of any on-line algorithm

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

On-line 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 can not be canceled or rerouted
- Permanent chains



On-line Algorithm: ACE

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

Idea: cost for hosting NF for a chain: 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 μ = 2L + 2

On-line Algorithm: ACE

Algorithm ACE:

- When request σ_j arrives,
  check if there exists a chain c_j , s.t.
  1. σ_j can be routed through c_j on a path of length at most R and
  2. $\sum_{v \in c_j} \frac{w_v(j)}{\kappa(v)} \leq L$
- If such c_j exists, then admit σ_j and assign it to c_j.
  Otherwise, reject σ_j.

On-line Algorithm: ACE

Theorem: Assume, $\min_v(\kappa(v)) \geq \log \mu$. Then ACE

- never violates capacity and length constraints and
- is $O(\log L)$ competitive.

Proof sketch:

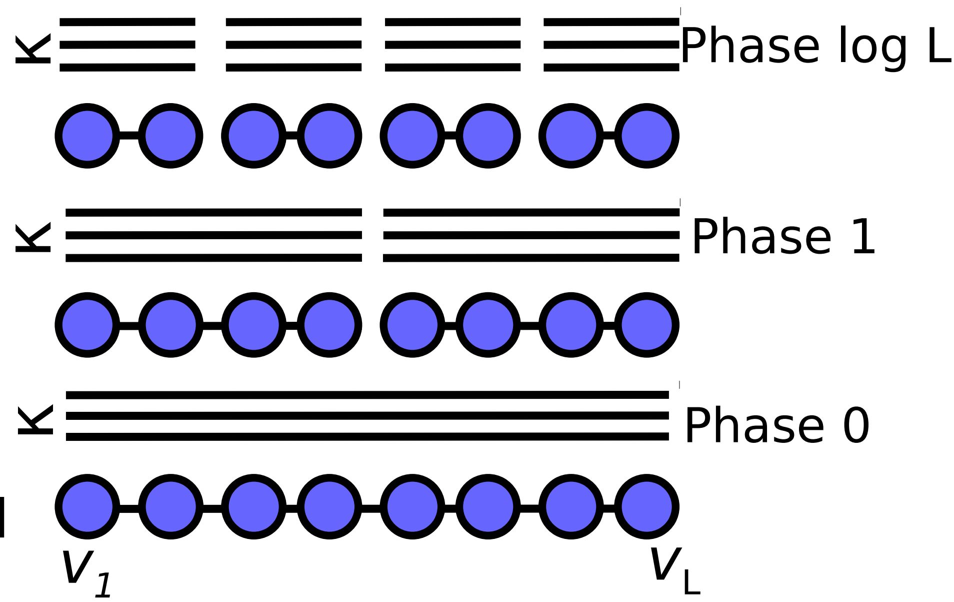
- Set of requests admitted by ACE respects constraints.
- W: sum of node costs,
  |A|: # requests admitted by ACE.
  At any moment, $W \leq |A| \cdot O(L \cdot \log \mu)$.
- |A*|: # requests admitted by the optimal offline algorithm but rejected by ACE. Then $|A^*| \leq W / L$.
- $|OPT| \leq |A| + |A^*| \leq |A| + |W| / L$
  $\leq |A| + |A| \cdot O(L \cdot \log \mu) / L$
  $= |A|\, O(\log \mu)$.

Lower bound on the Competetive Ratio

Theorem: Assume, $\kappa \geq \log \mu$. Any on-line algorithm for SCEP must have a competitive ratio of at least $\Omega(\log L)$.

Proof sketch:

- $c=(v_1, \ldots, v_L)$
- Different chains overlap at c.
- Requests in log L + 1 phases
- Phase i: $2^i \kappa$ requests
- Adversary stops sending requests after a phase j, when the on-line algorithm admitted at most $2^{j+1} \kappa / \log L$ requests until phase j. Such a j must exist.
- Optimal offline algorithm rejects all requests except the $2^j \kappa$ requests of phase j.



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

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 Integer Linear Program – NP-completeness

Exact optimal solution via 0-1-ILP

$$\text{maximize} \quad \sum_{\sigma_i \in \sigma} x_i \tag{1}$$

$$\text{s.t.} \quad x_i - \sum_{c \in \mathcal{C}} x_{c,i} = 0 \qquad \forall\ \sigma_i \in \sigma \tag{2}$$

$$\sum_{c \in \mathcal{C} : \sigma_i \notin S_c} x_{c,i} = 0 \qquad \forall\ \sigma_i \in \sigma \tag{3}$$

$$x_c \leq x_v \qquad \forall\ v \in V, \forall\ c \in \mathcal{C} : v \in c \tag{4}$$

$$\sum_{c \in \mathcal{C} : v \in c} x_c \geq x_v \qquad \forall\ v \in V \tag{5}$$

$$\sum_{\sigma_i \in \sigma} \sum_{c \in \mathcal{C} : v \in c} x_{c,i} \leq \kappa(v) \cdot x_v \qquad \forall\ v \in V \tag{6}$$

$$x_i, x_v, x_c, x_{c,i} \in \{0, 1\} \qquad \forall\ v \in V, \forall\ c \in \mathcal{C}, \forall\ \sigma_i \in \sigma \tag{7}$$

Summary

- Trend: Network Function Virtualization
- First step towards a better understanding of the algorithmic problem
- First algoritm with quality guarantee
- Main contributions:
  - O(log L)-competetive on-line algorithm
  - Ω(log L) lower bound on the competetive ratio of any on-line algorithm
  - Offline SCEP:
    - APX-hard for unit capacities and constant $L \geq 3$
    - Poly-APX-hard, when there is no bound on L
    - Exact optimal solution via 0-1-ILP
    - NP-completeness for constant L

Thank you!