Network Abstraction with Provisioning Guarantees for Multi-Domain VNE

Yanis Achaichia

Supervised by Christelle CAILLOUET, Nicolas HUIN
and Géraldine TEXIER

Contents

Problem Introduction

Multi-Domain **Virtual Network Embedding**

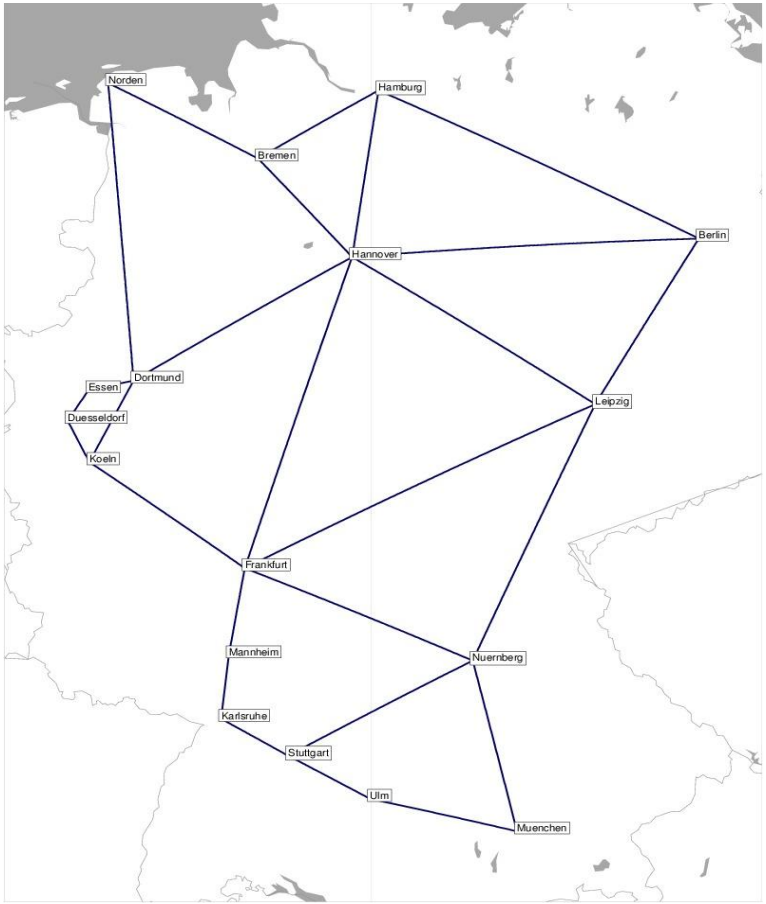
Problem Introduction

Multi-Domain **Virtual Network Embedding**

Physical Network:

(Directed) Graph G=(V,A)

- Set of node metrics
  - Qualitative: types of accepted functions
  - Quantitative: Available computing power
- Set of edge metrics
  - Additive: delay
  - Quantitative: maximum bandwidth



nobel_germany network; source: SNDlib

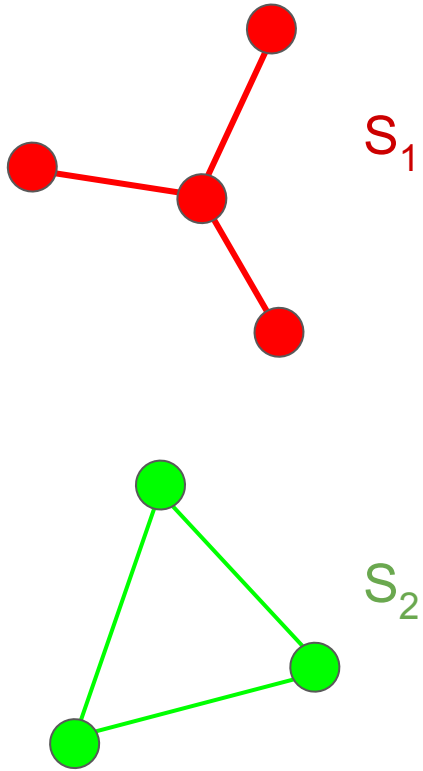
Problem Introduction

Multi-Domain **Virtual Network Embedding**

Network services (slices):

Directed graphs $S_k=(V_k', A_k')$

- Set of node requirements:
  - Qualitative: network function type
  - Quantitative: required computation power
- Set of edge requirements:
  - Additive: maximum accepted delay
  - Quantitative: required bandwidth



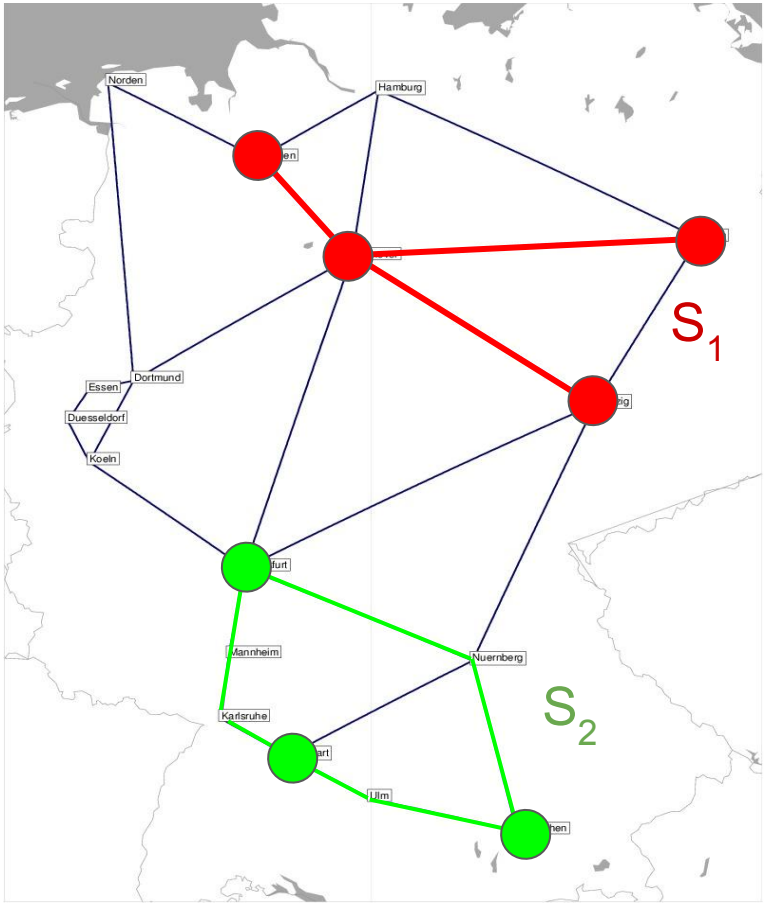
Problem Introduction

Multi-Domain **Virtual Network Embedding**

Goal:

Finding an assignment of the nodes and edges such that:

- Each node of the service graph corresponds to a node in the physical network
- Each each edge of the service graph corresponds to a path in the physical network
- All node and edge requirements are met by their corresponding metrics

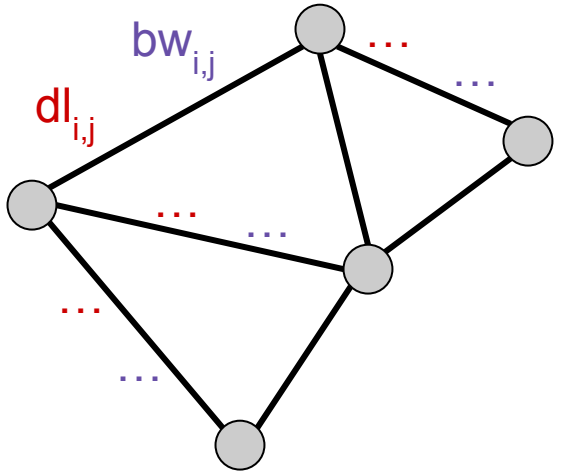


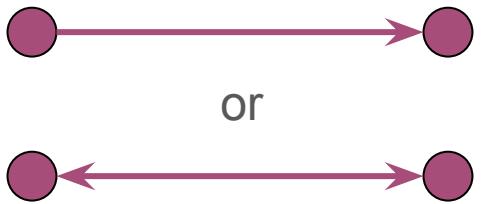
Problem Introduction

Simplified routing version

- Only edge metrics (bandwidth, delay)

- Simple 2-node slice request (with matching requirements)





Problem Introduction

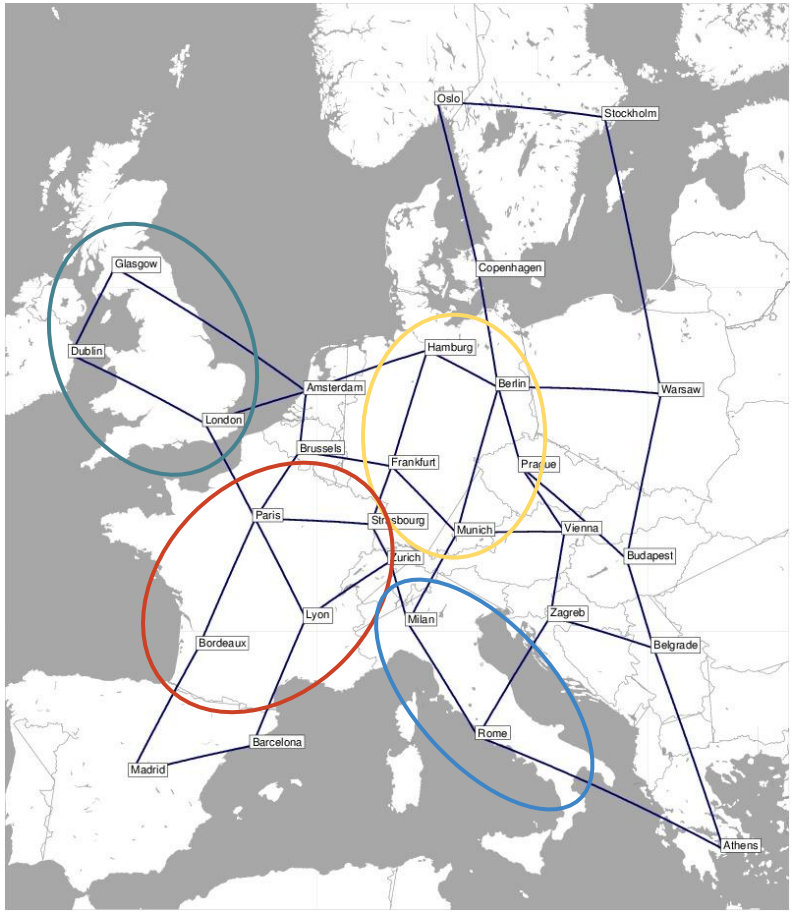
Multi-Domain Virtual Network Embedding

Problem Introduction

Multi-Domain Virtual Network Embedding

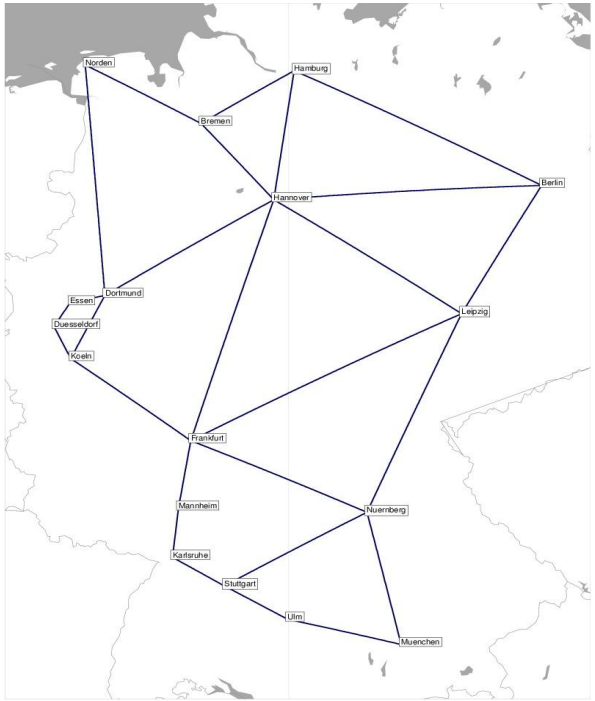
The problem scales to a large network divided into of multiple administrative domains

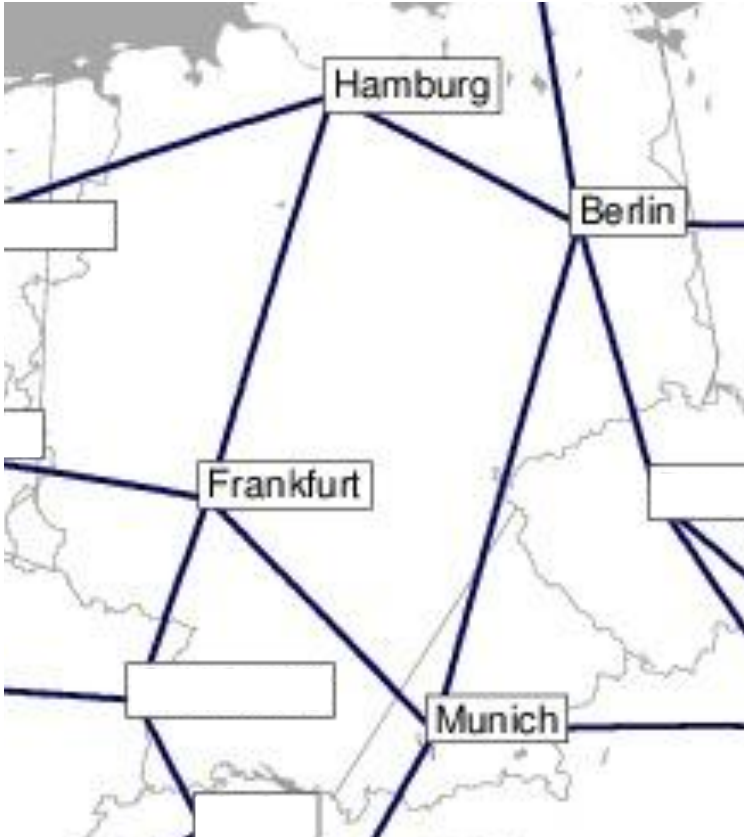
Each domain is an *Infrastructure Provider* (InP) that can receive and deal out service embedding requests.



nobel_eu network; source: SNDlib

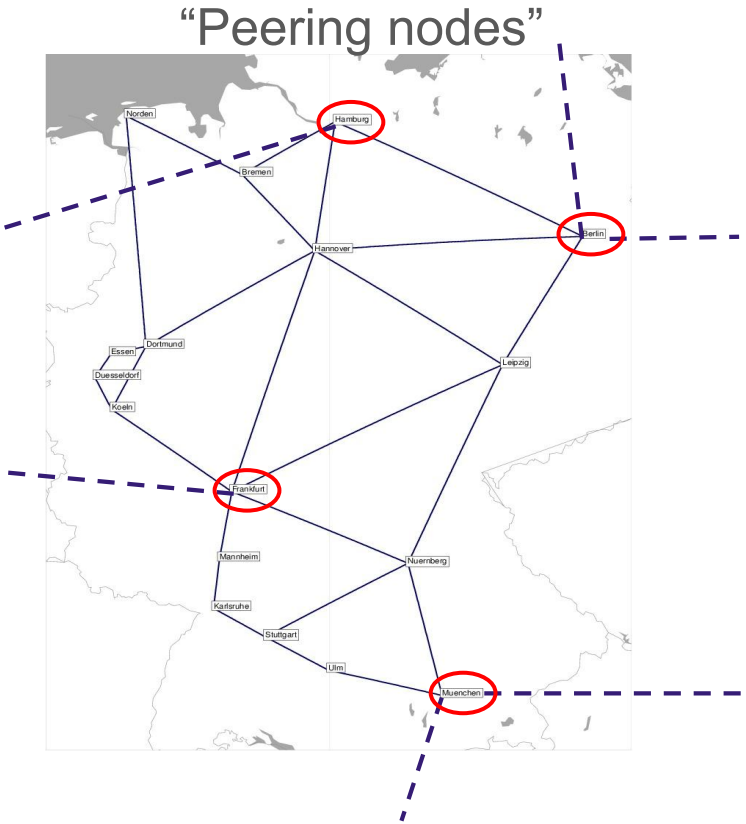
Problem Introduction

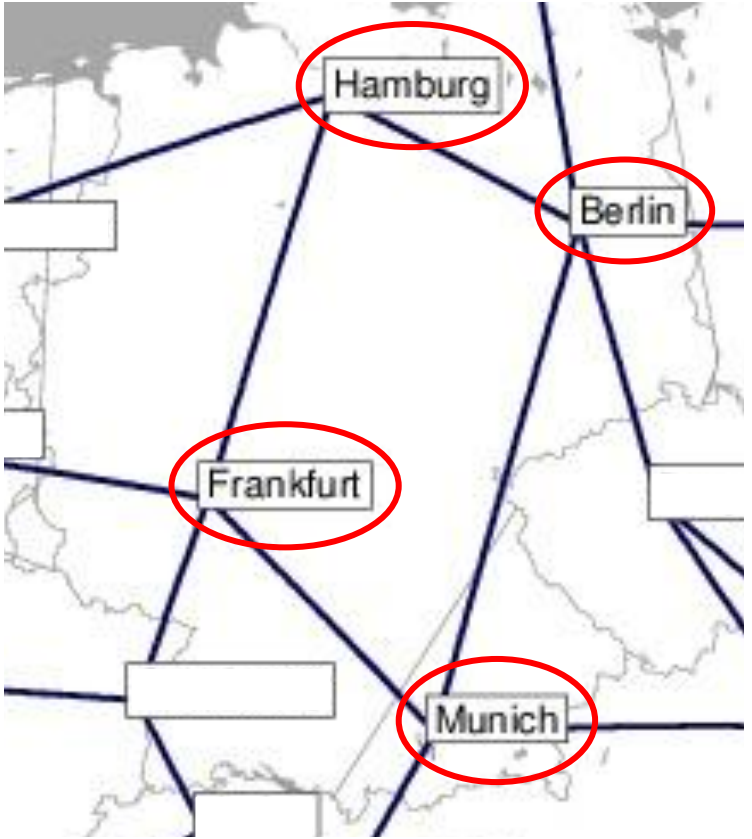




nobel_eu network; source: SNDlib

Problem Introduction





nobel_eu network; source: SNDlib

Problem Introduction

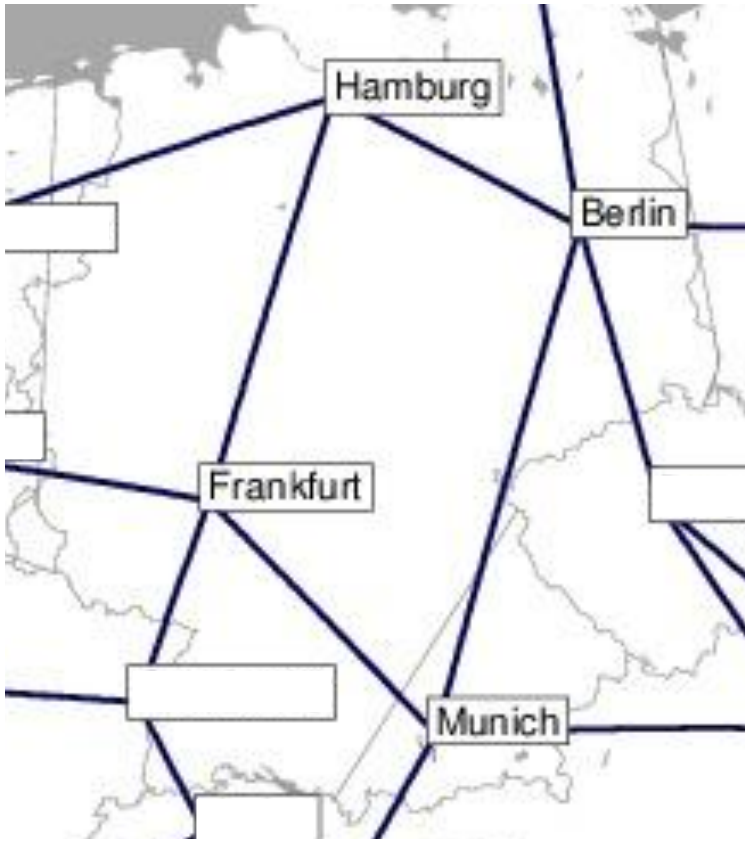
Multi-Domain Virtual Network Embedding

Public information:

- Each domain's peering nodes
- Inter-domain links

Private information:

- Non-peering nodes
- Intra-domain edges



nobel_eu network; source: SNDlib

Problem Introduction

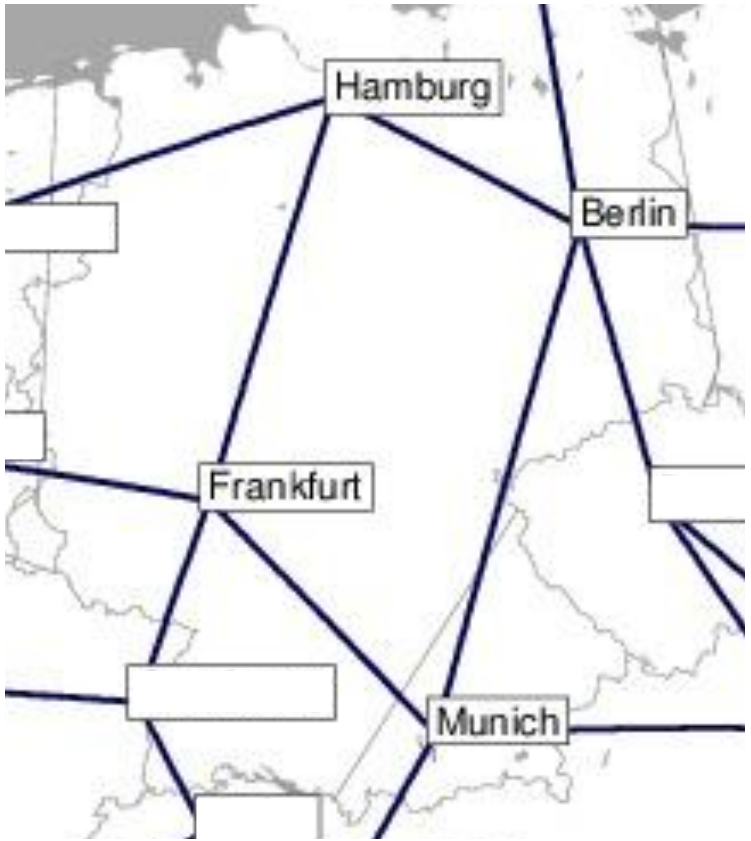
Multi-Domain Virtual Network Embedding

Public information:

- Each domain's peering nodes
- Inter-domain links

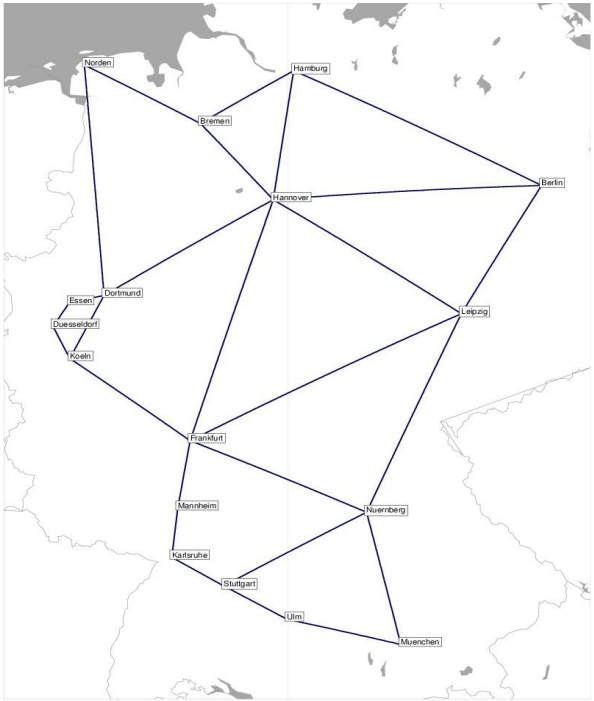
Private information: Sensitive information

- Non-peering nodes
- Intra-domain edges



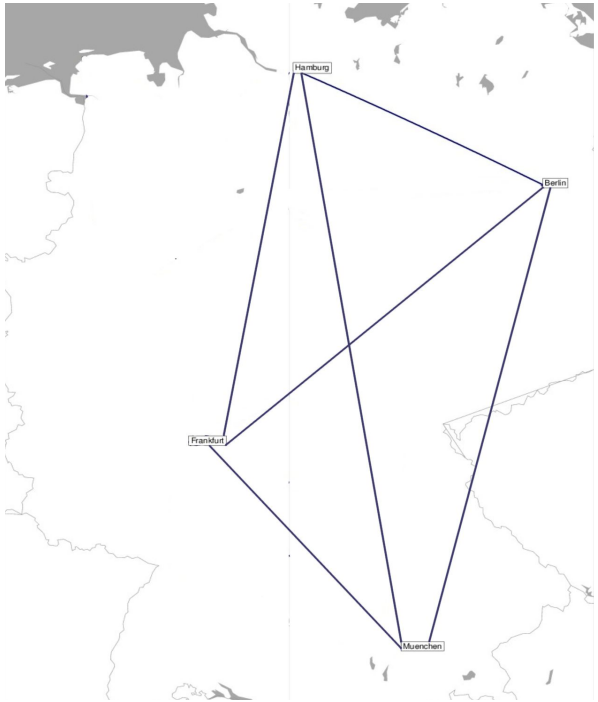
nobel_eu network; source: SNDlib

Goal of the work



Provide a methodology for network abstraction

That behaves well for the VNE problem



Contents

- Context and Problem Introduction
- Provisioning properties of an abstraction
- Abstraction workflow
- Results
- Future work

The value of the under-provisioning guarantee

Under-provisioning (advertising strictly fewer resources)**:**

A network abstraction is said to be *under-provisioned* if all service requests that can be embedded on it can also be accepted on the original network

$$\{S: S \prec G'\} \subseteq \{S: S \prec G\}$$

The value of the under-provisioning guarantee

Under-provisioning (advertising strictly fewer resources)**:**

A network abstraction is said to be *under-provisioned* if all service requests that can be embedded on it can also be accepted on the original network

$$\{S: S \prec G'\} \subseteq \{S: S \prec G\}$$

Over-provisioning (advertising strictly more resources)**:**

A network abstraction is said to be *over-provisioned* if all service requests that can be embedded on it can also be accepted on the original network

$$\{S: S \prec G\} \subseteq \{S: S \prec G'\}$$

The value of the under-provisioning guarantee

Under-provisioning (advertising strictly fewer resources)**:**

A network abstraction is said to be *under-provisioned* if all service requests that can be embedded on it can also be accepted on the original network

$$\{S: S \prec G'\} \subseteq \{S: S \prec G\}$$

Minimal messaging overhead

Over-provisioning (advertising strictly more resources)**:**

A network abstraction is said to be *over-provisioned* if all service requests that can be embedded on it can also be accepted on the original network

$$\{S: S \prec G\} \subseteq \{S: S \prec G'\}$$

Interesting case

Notably, if G' is an under-provisioned and over-provisioned abstraction:

$$\{S: S \prec G'\} = \{S: S \prec G\}$$

$\Rightarrow$ "perfect" representation of the original network

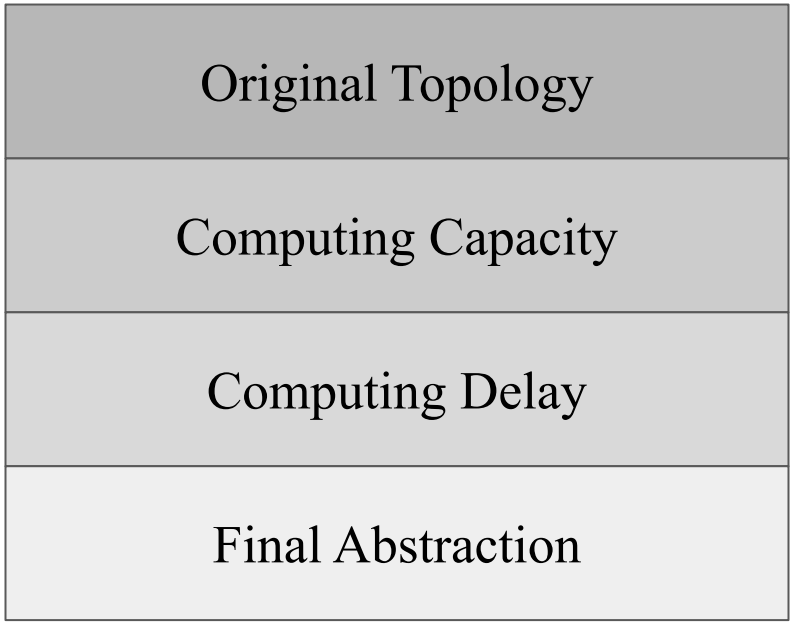
⚠️ Raises questions about privacy

Contents

- Context and Problem Introduction
- Provisioning properties of an abstraction
- Abstraction workflow
- Results
- Future work

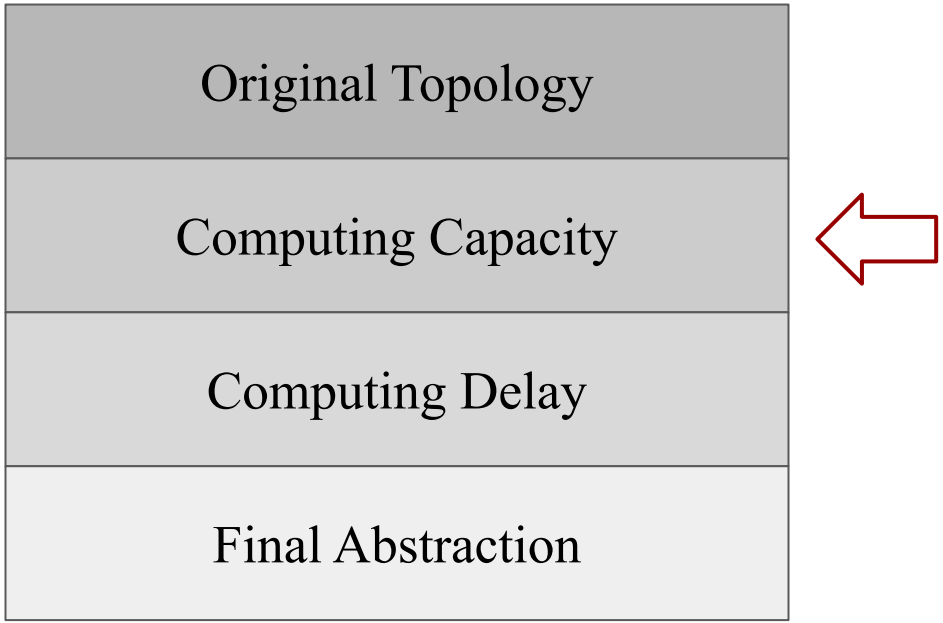
Global view of the process

- “Legacy” abstraction style
- Separately computed metrics
- Under-provisioning guarantee



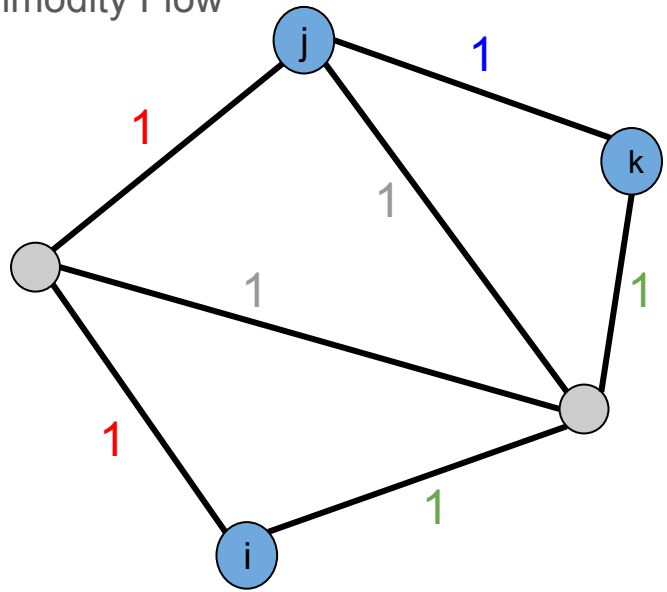
Global view of the process

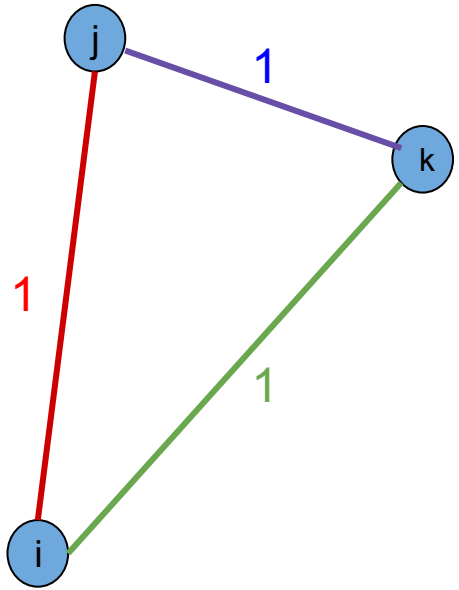
- “Legacy” abstraction style
- Separately computed metrics
- Under-provisioning guarantee



Part 1: Determining the capacity

Multi-Commodity Flow

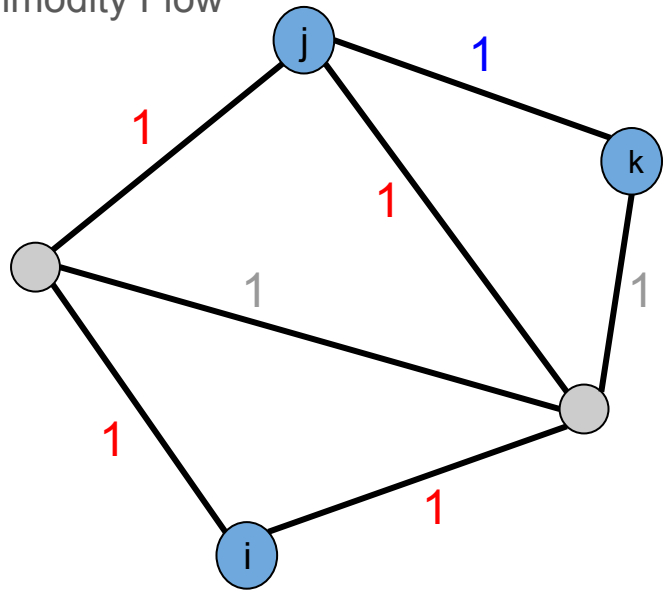


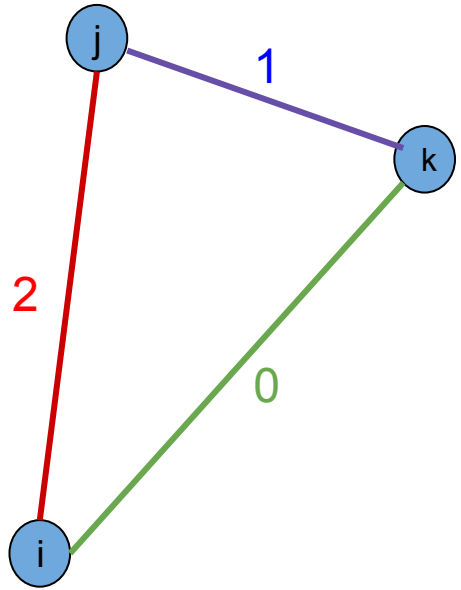


Takes into account the interactions between pairs

Part 1: Determining the capacity

Multi-Commodity Flow

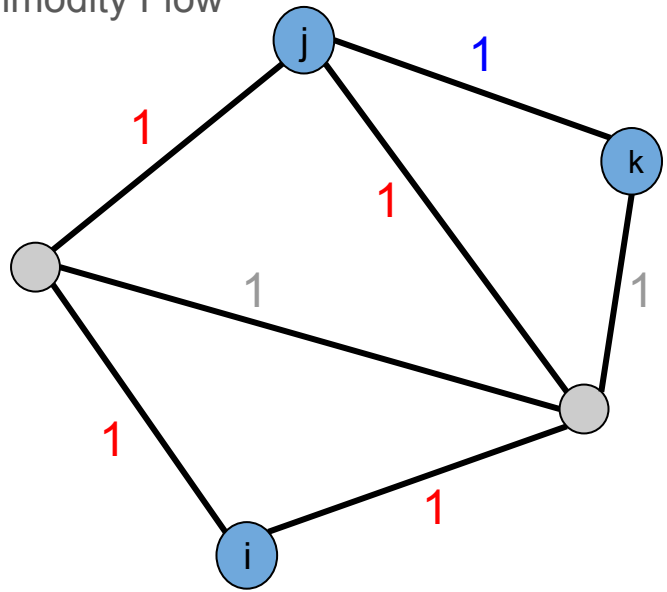


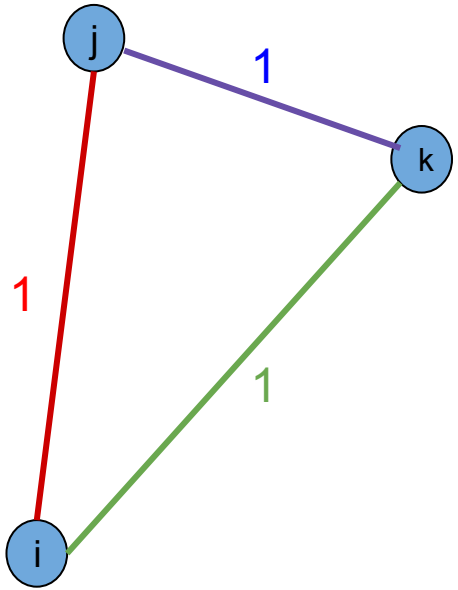


How to choose the right way of advertising bandwidth

Part 1: Determining the capacity

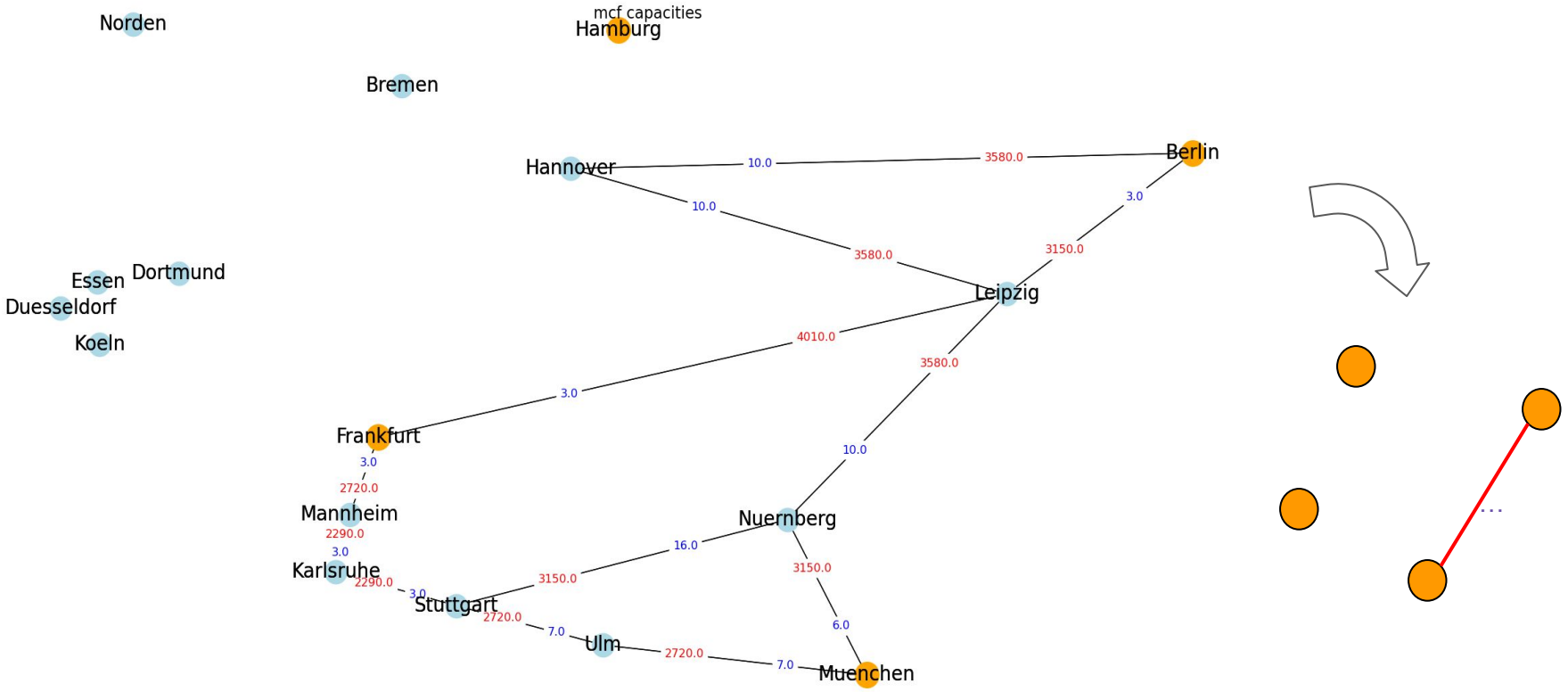
Multi-Commodity Flow



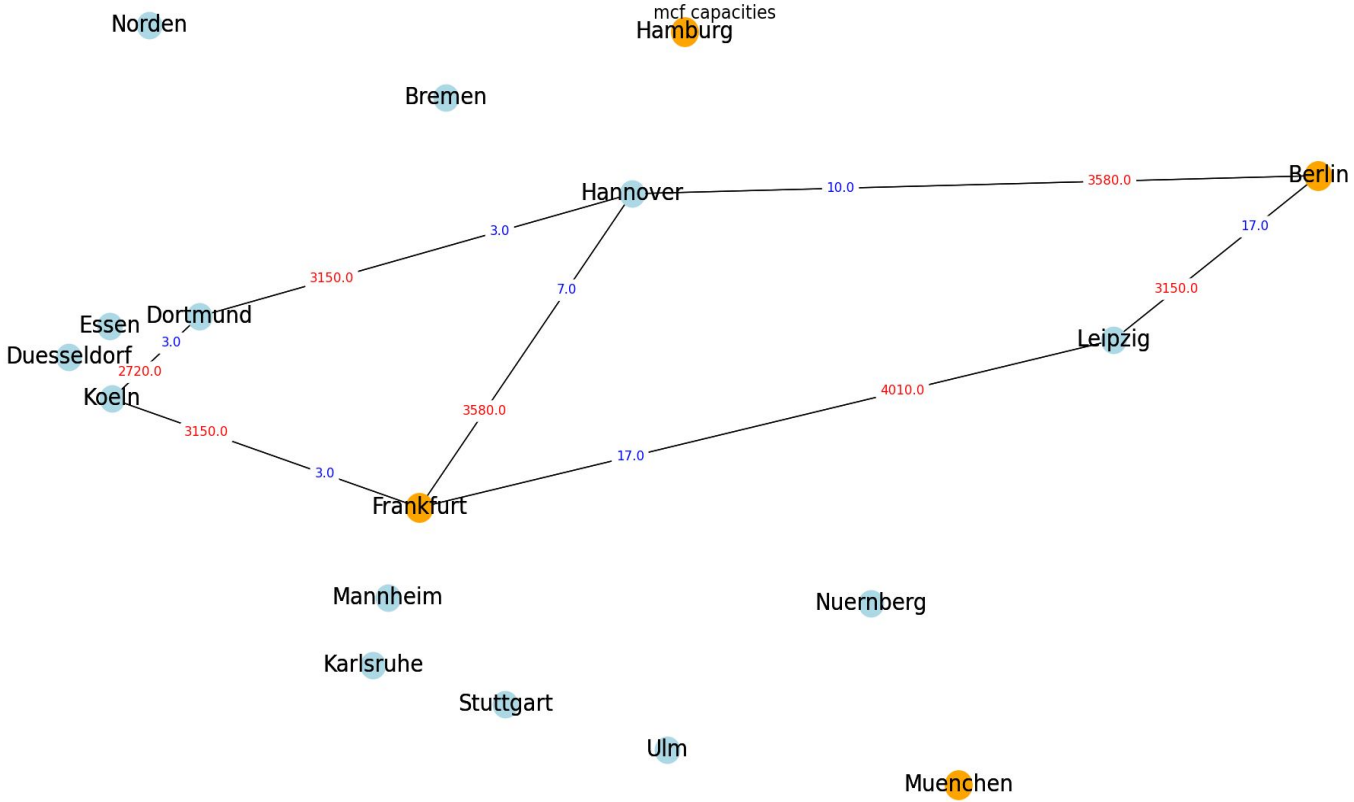


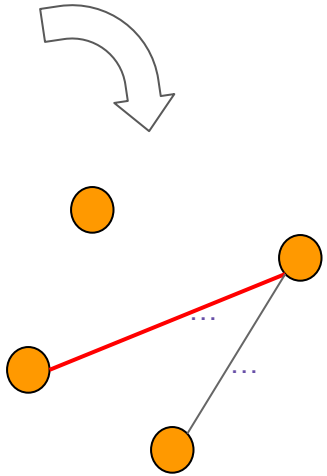
How to choose the right way of advertising bandwidth

“Commodity subgraphs”

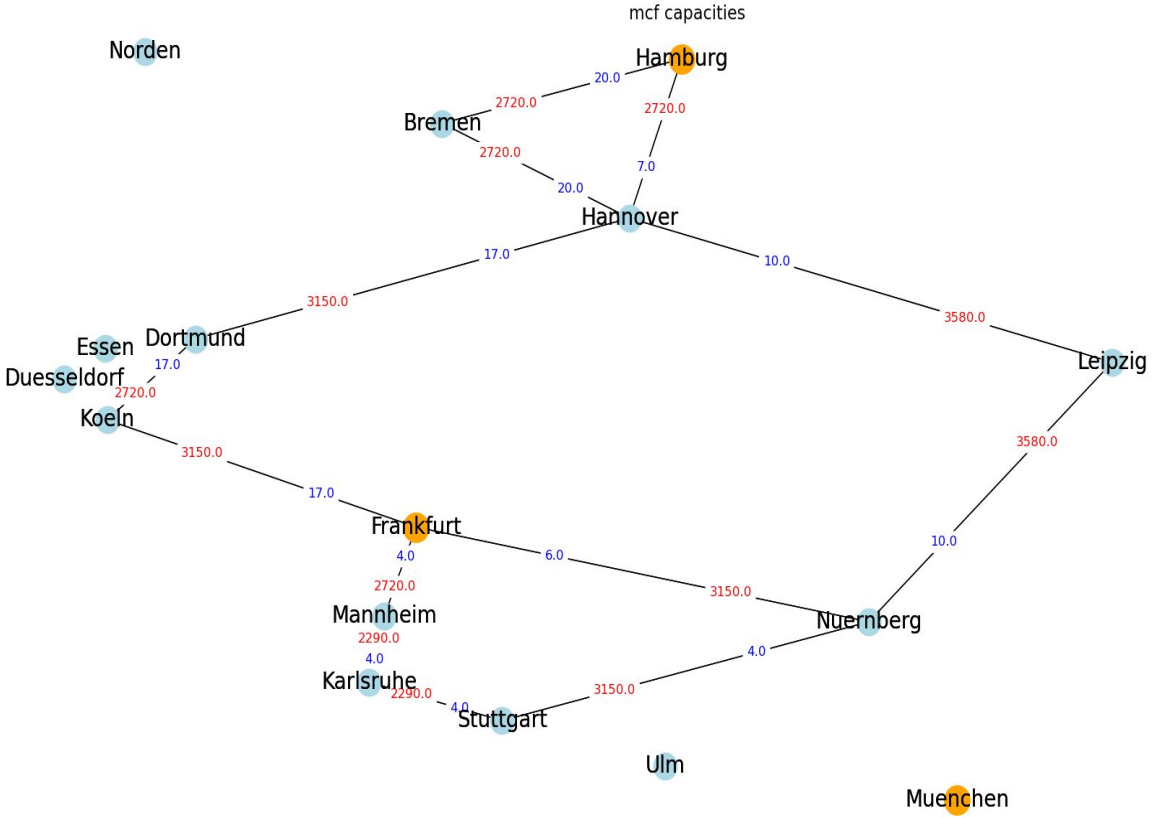


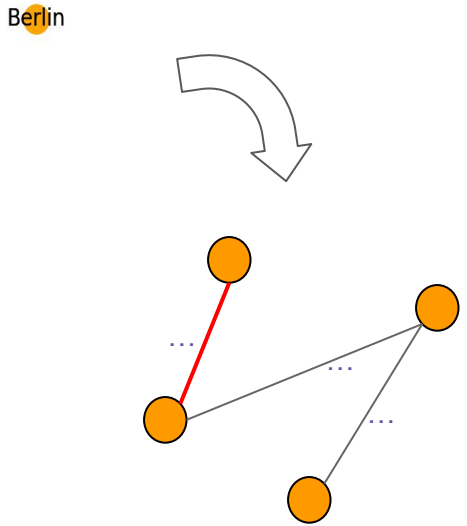
“Commodity subgraphs”



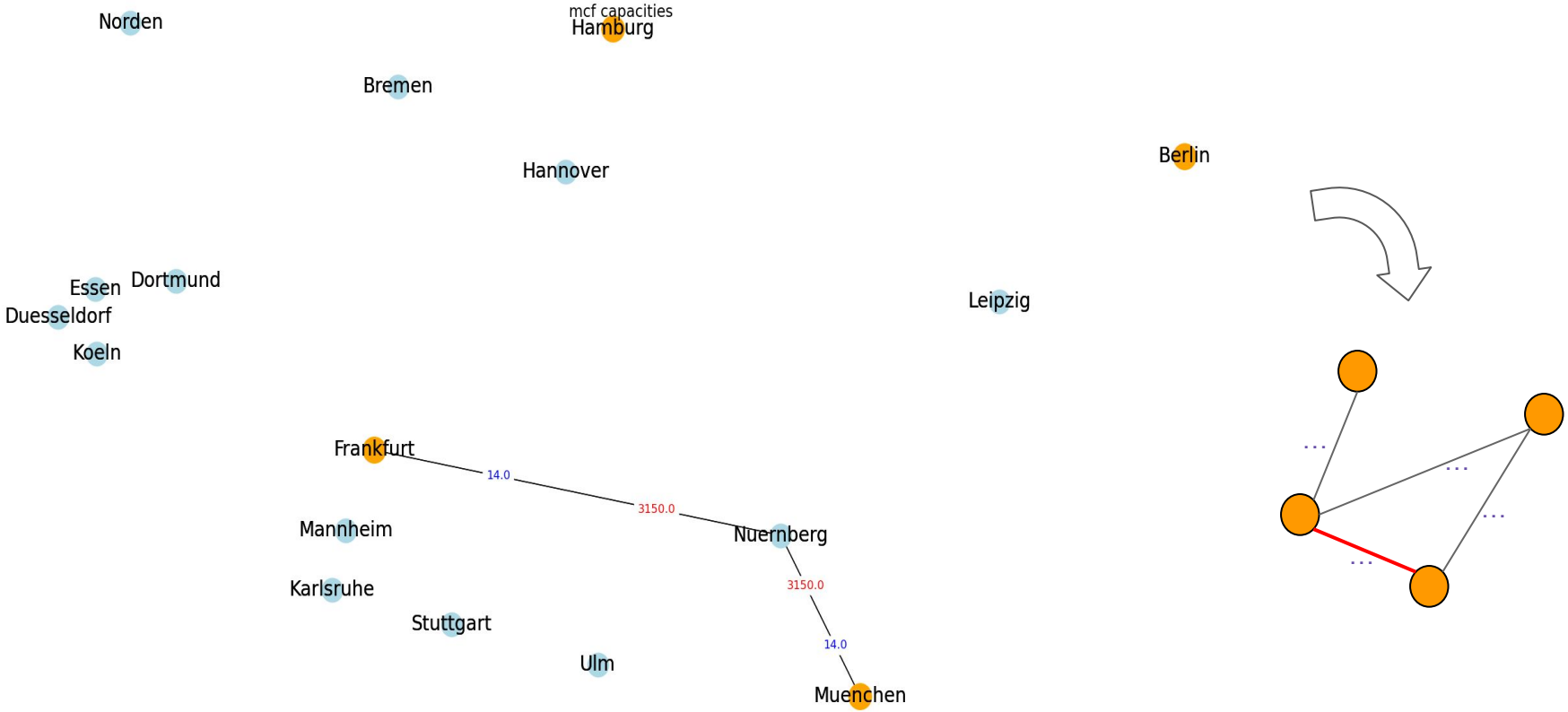


“Commodity subgraphs”

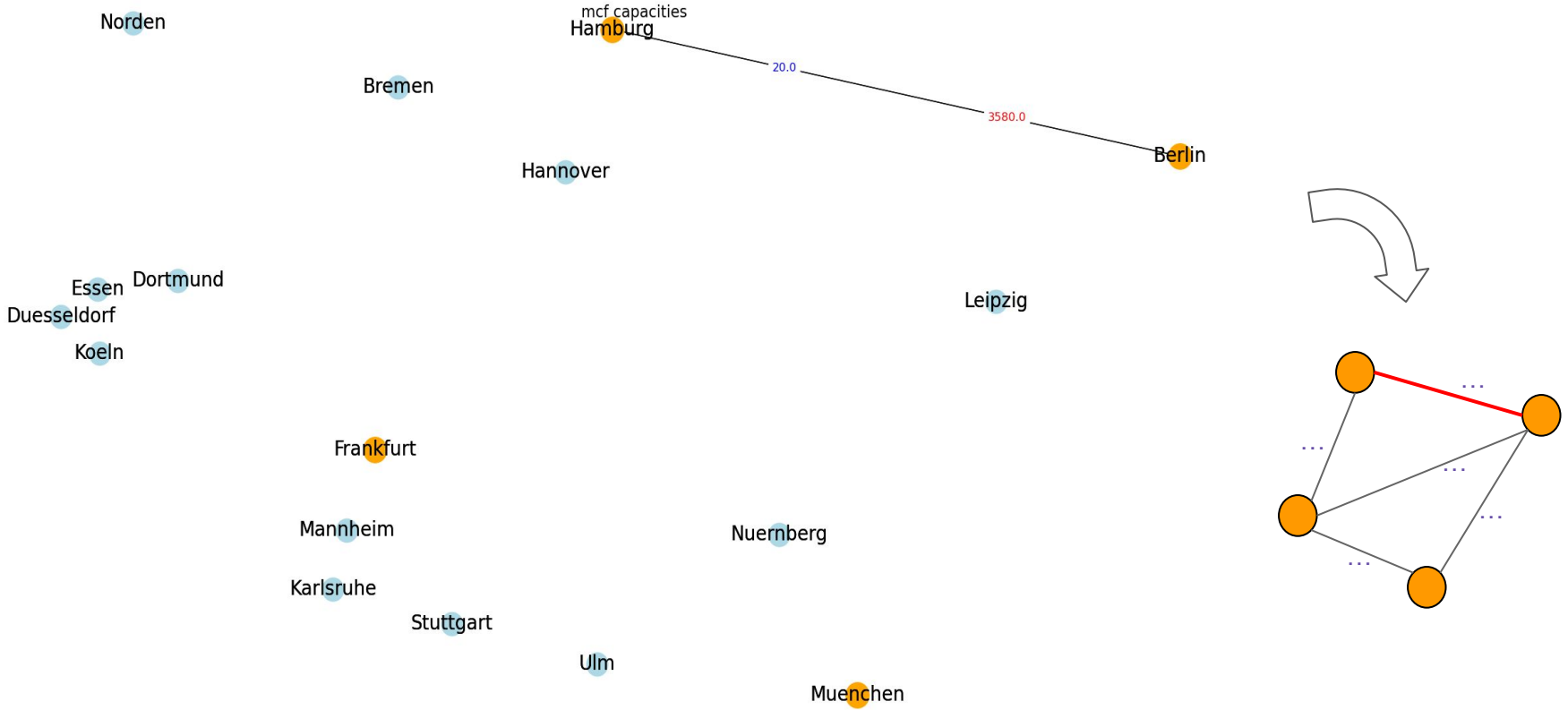




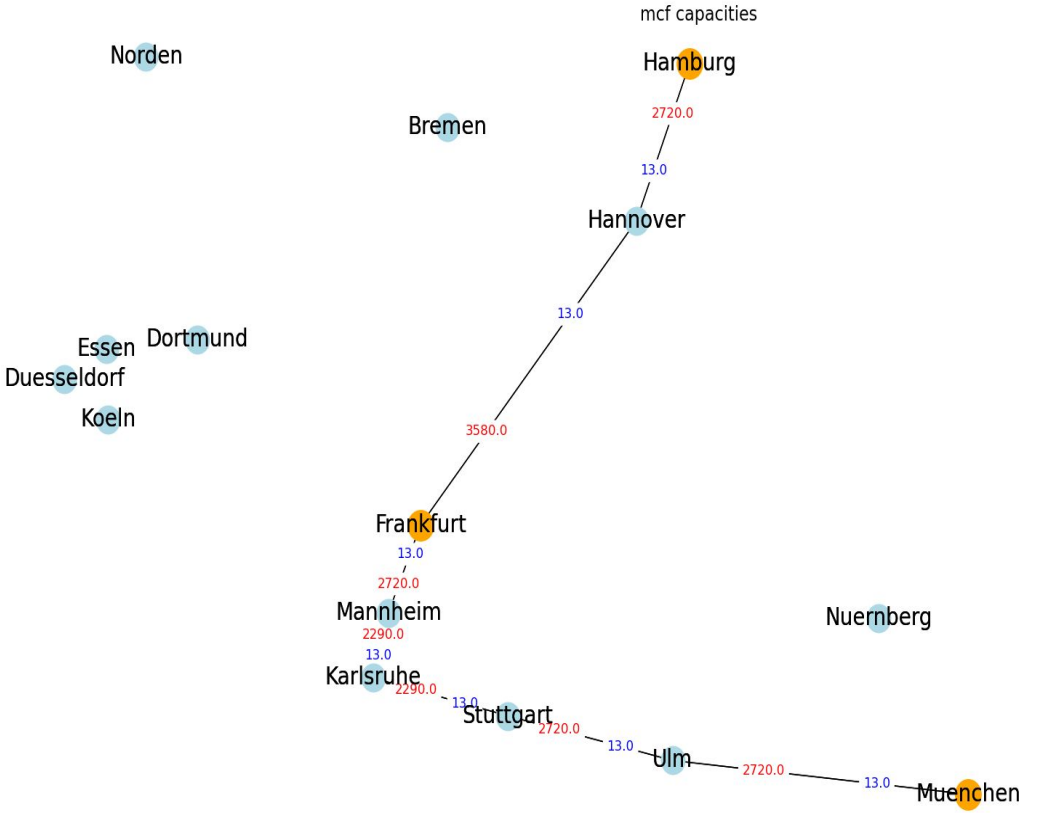
“Commodity subgraphs”

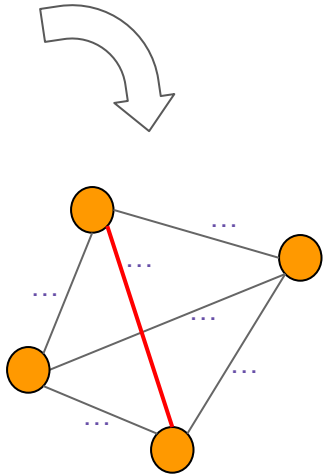


“Commodity subgraphs”



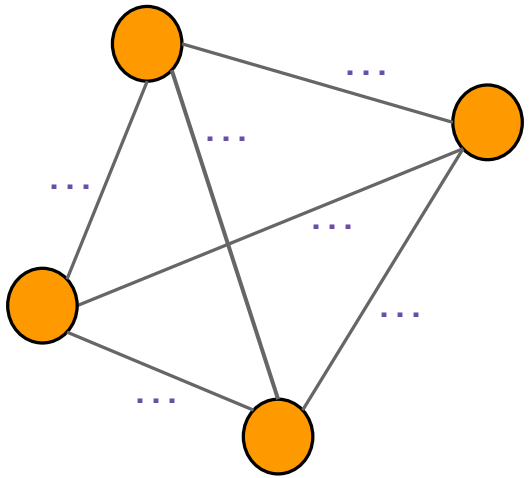
“Commodity subgraphs”





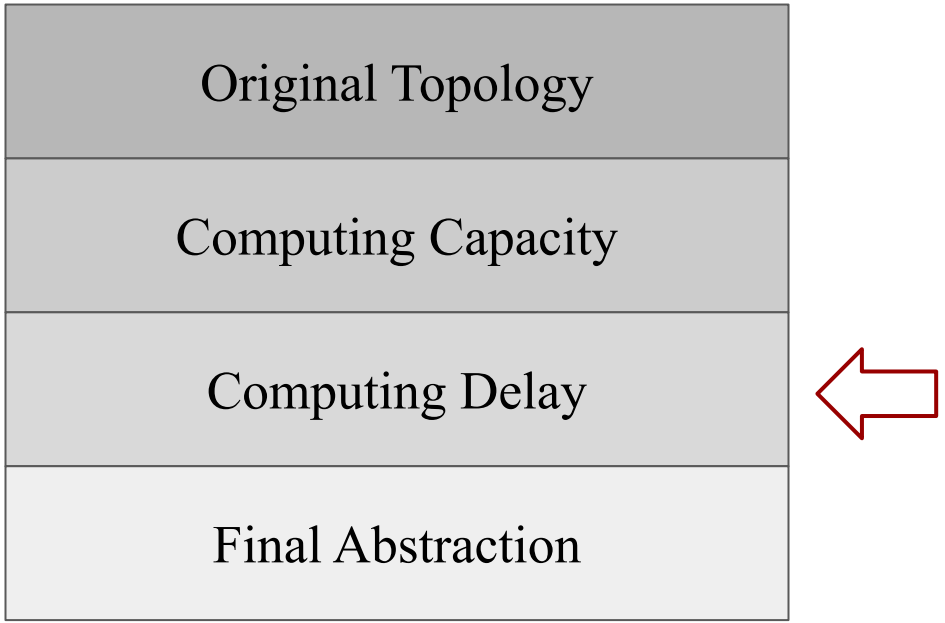
First step of the abstraction

Each abstracted edge now has a capacity



Global view of the process

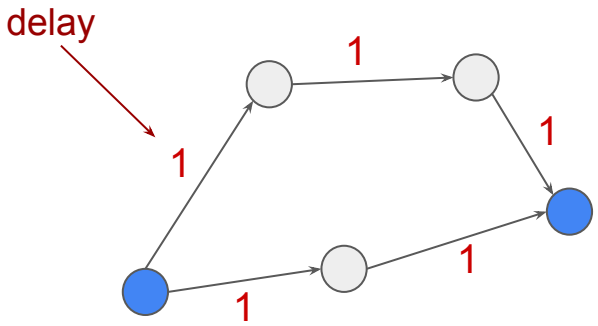
- “Legacy” abstraction style
- Separately computed metrics
- Under-provisioning guarantee



The need for an adaptive delay

Idea: Resulting delay of an embedding depends on the bandwidth usage

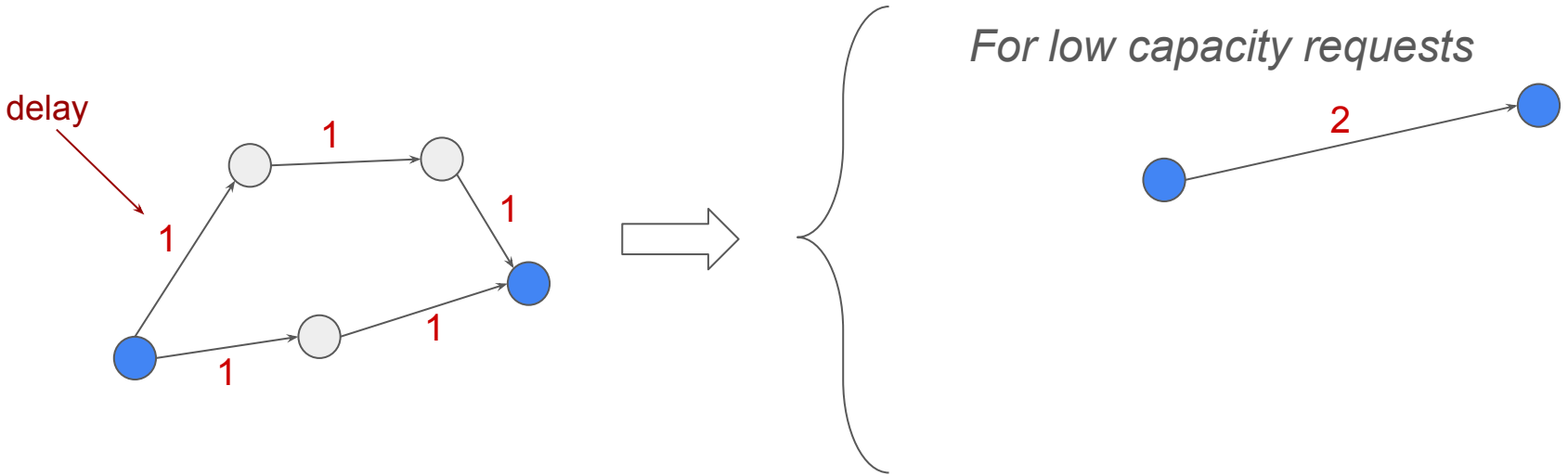
(as a single abstracted edge represents multiple paths)



The need for an adaptive delay

Idea: Resulting delay of an embedding depends on the bandwidth usage

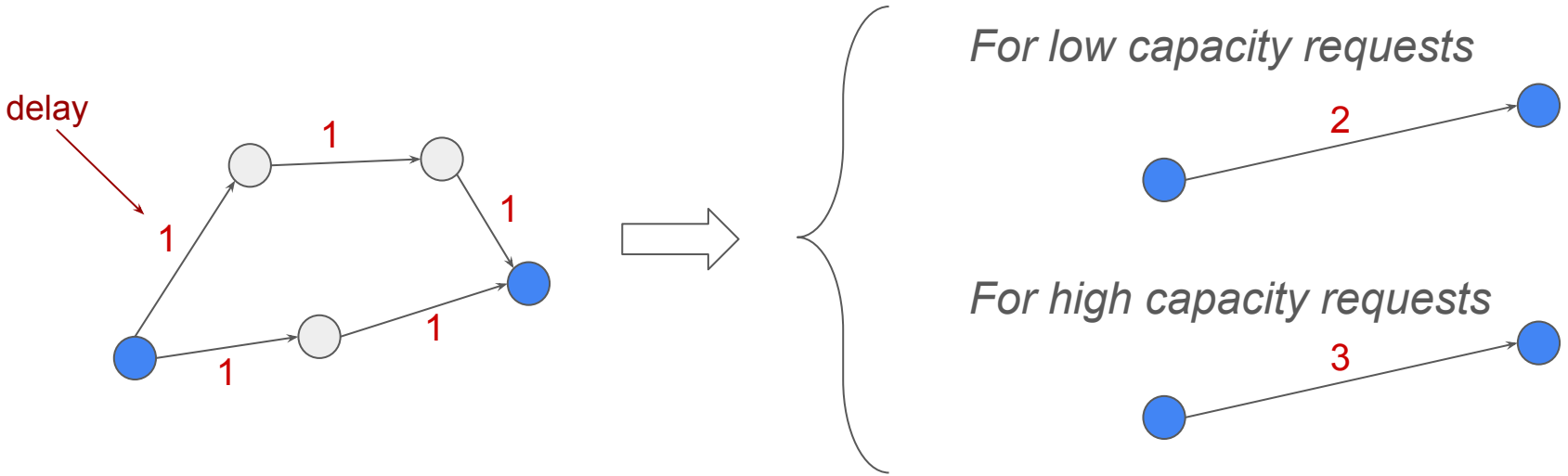
(as a single abstracted edge represents multiple paths)



The need for an adaptive delay

Idea: Resulting delay of an embedding depends on the bandwidth usage

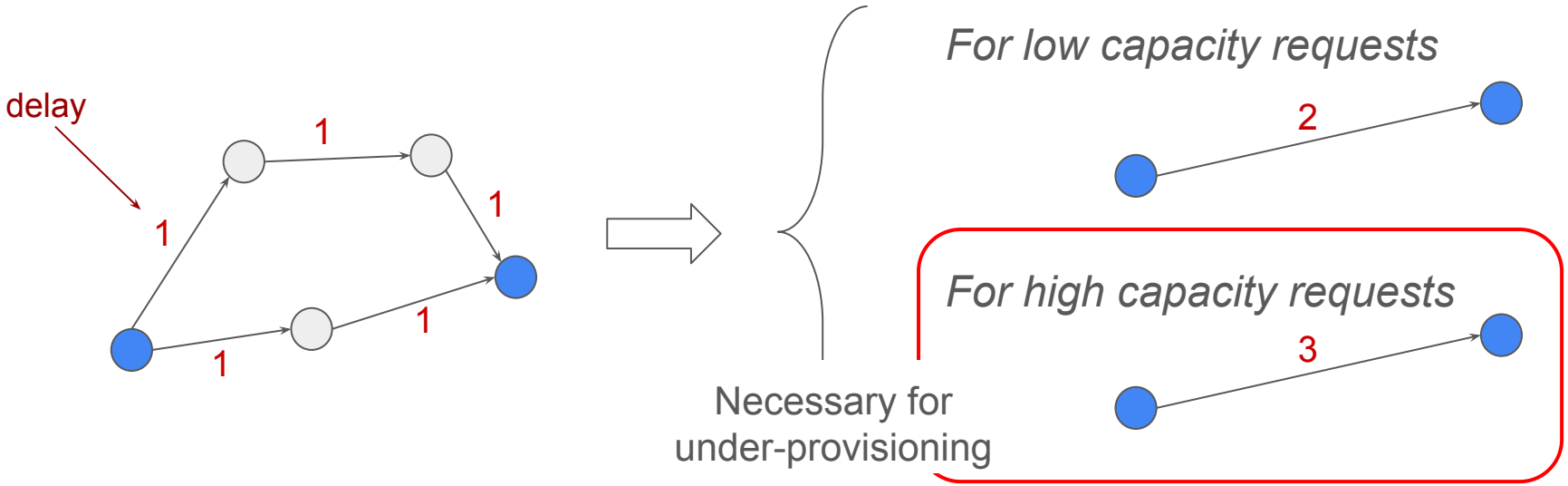
(as a single abstracted edge represents multiple paths)



The need for an adaptive delay

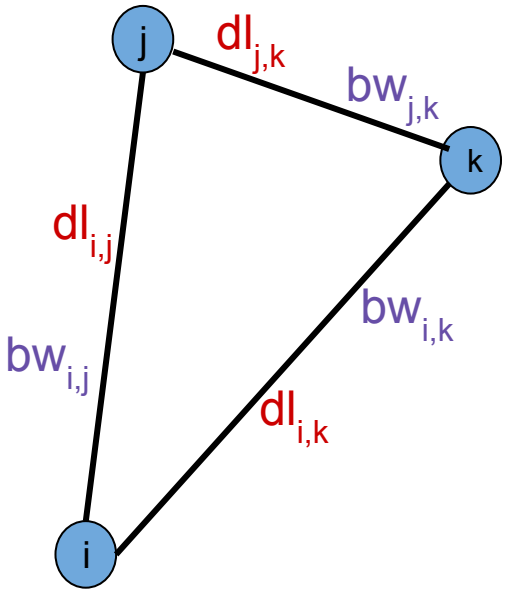
Idea: Resulting delay of an embedding depends on the bandwidth usage

(as a single abstracted edge represents multiple paths)



Flow-delay illustrated

Instead of a single fixed abstraction

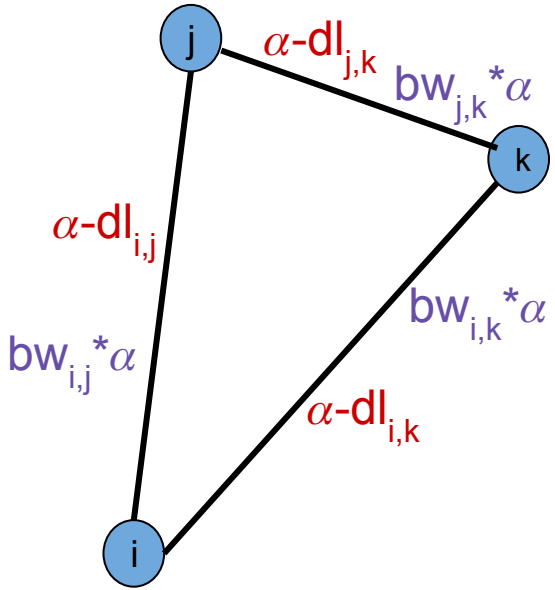


Flow-delay illustrated

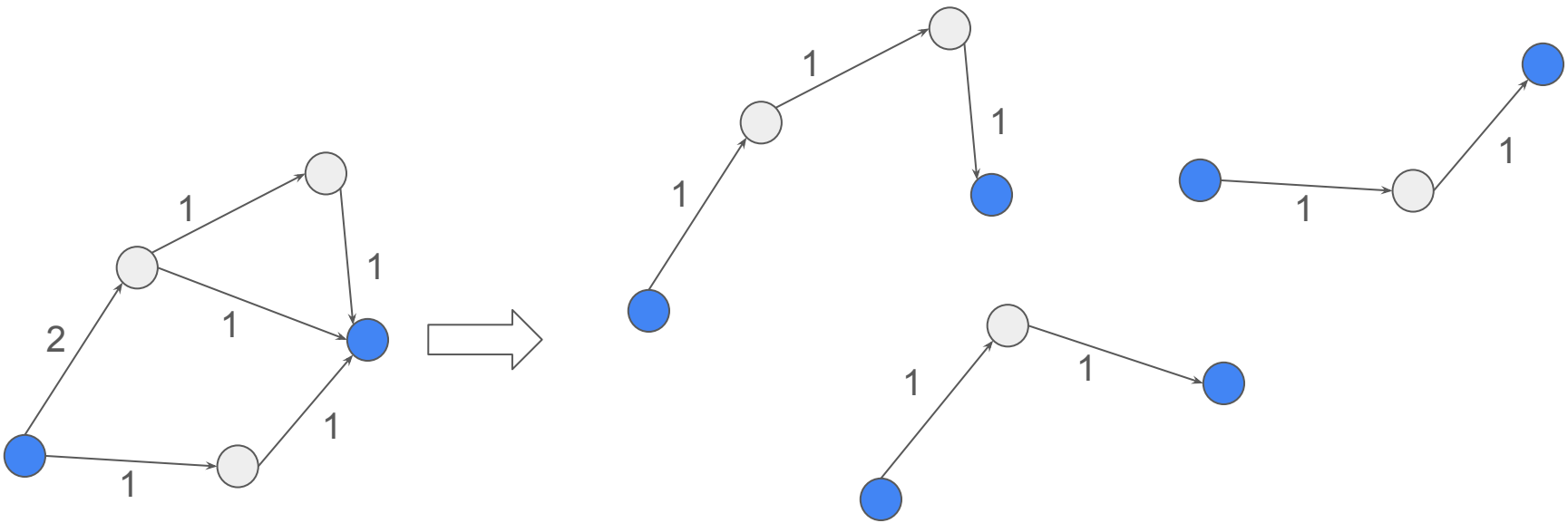
Instead of a single fixed abstraction

Create a group of abstractions dependent on a scalar $\alpha \in [0,1]$

Parametrize bandwidth to allow for better flow advertising



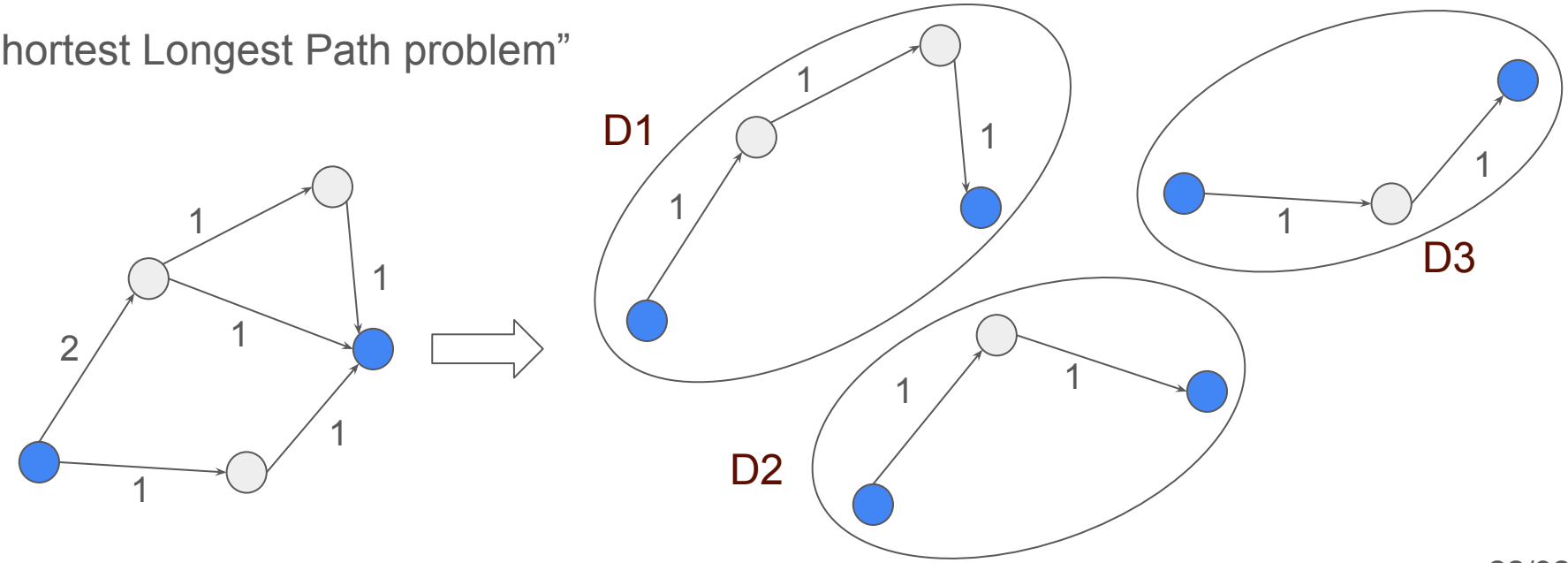
The link to flow decomposition



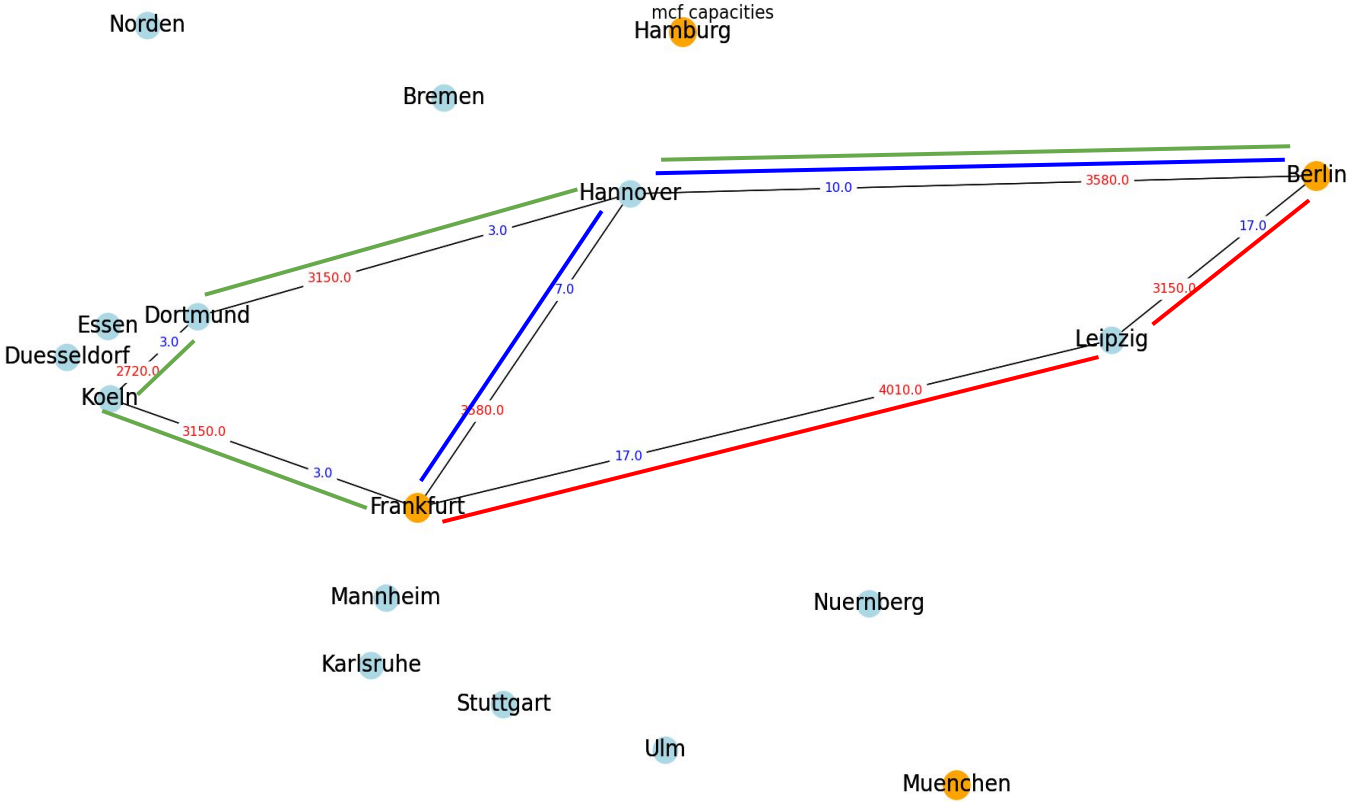
The link to flow decomposition

Minimising the delay of the longest decomposed path: min(max({D1, D2, D3}))

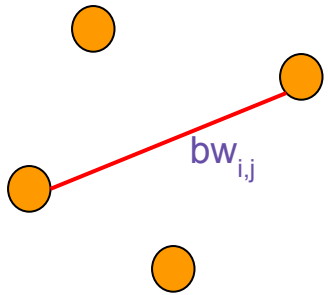
“Shortest Longest Path problem”



Decomposing a commodity subgraph



Chose the k-shortest paths that satisfy the bandwidth requirements (α*$bw_{i,j}$)

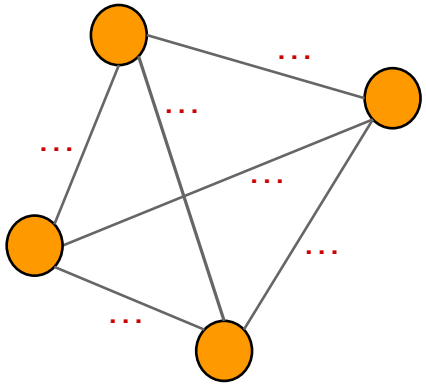


Extracting a delay value

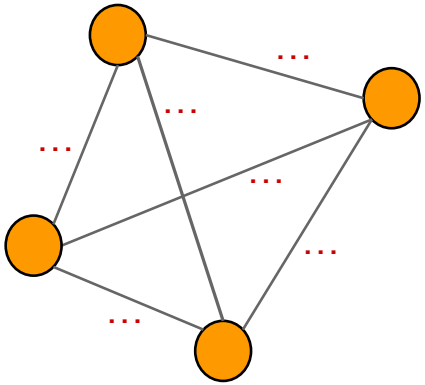
Run the algorithm on each abstracted edge

Delay := longest path of the decomposition

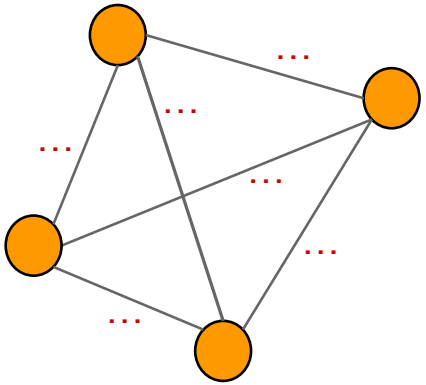
α = 0.2



α = 0.4



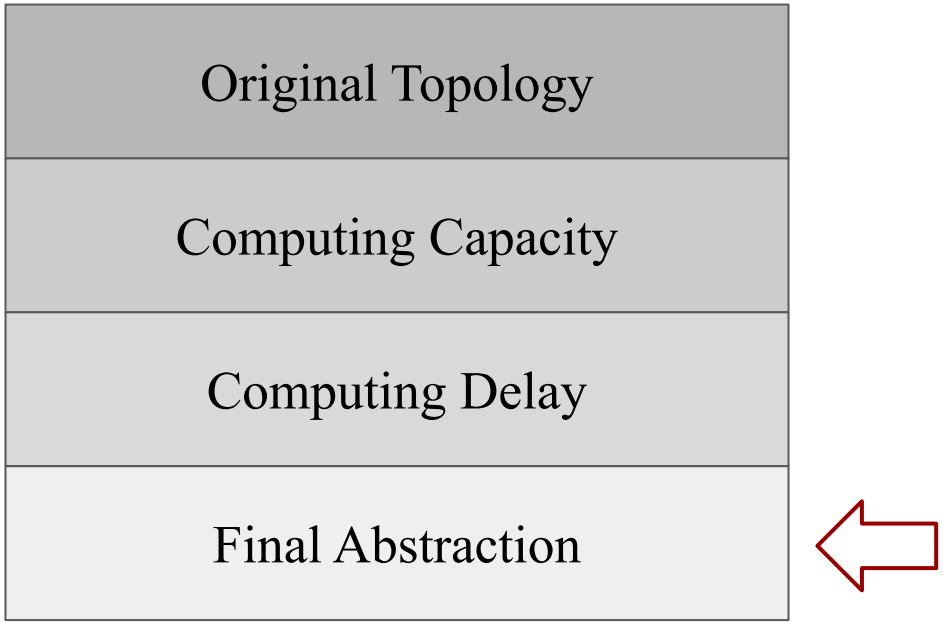
α = 0.6



…

Global view of the process

- “Legacy” abstraction style
- Separately computed metrics
- Under-provisioning guarantee



Contents

Main testing metric

Main testing metric:

$$\text{Service Acceptance (S.A.)} = \frac{\text{\# of embedded requests on G' (solution)}}{\text{\# of embedded requests on G (baseline)}}$$

Main testing metric

Main testing metric:

$$\text{Service Acceptance (S.A.)} = \frac{\text{\# of embedded requests on G' (solution)}}{\text{\# of embedded requests on G (baseline)}} \geq$$

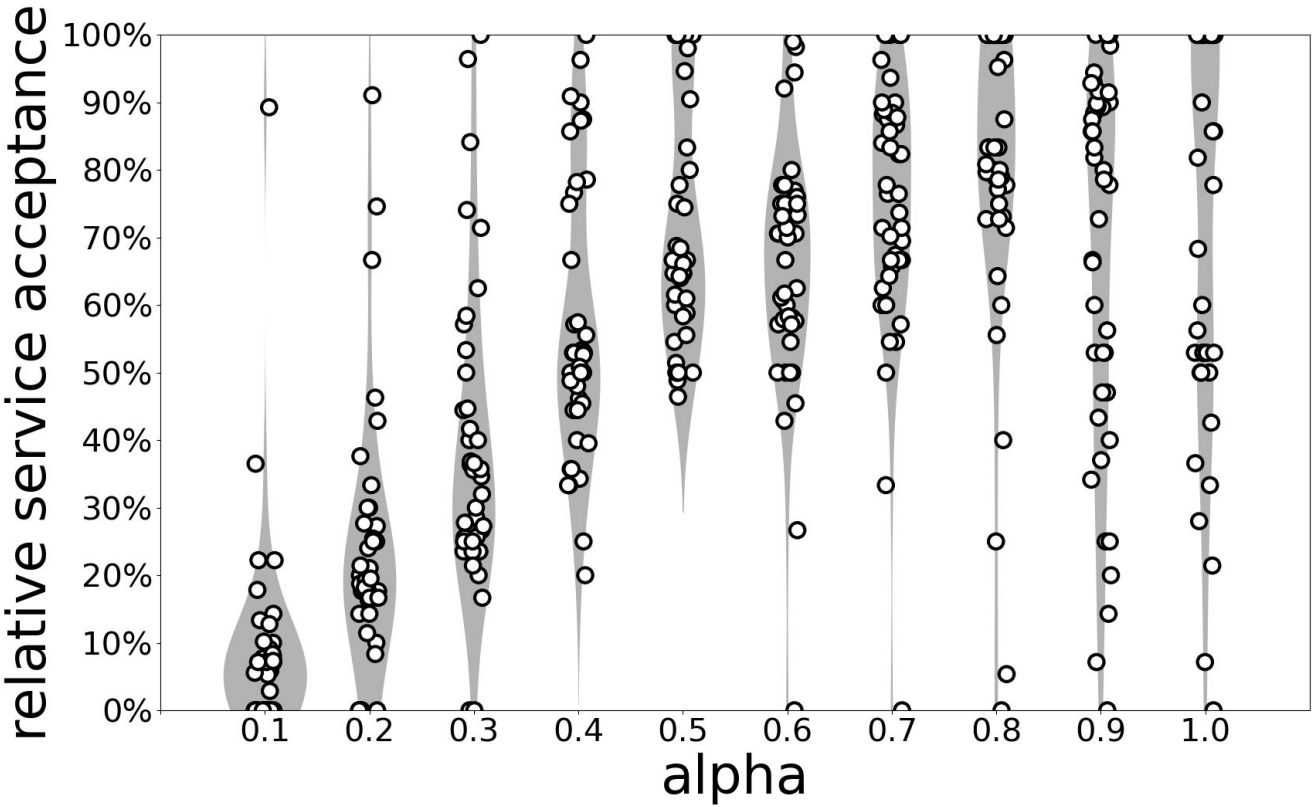
under-provisioning

$$S.A. \in [0,1]$$

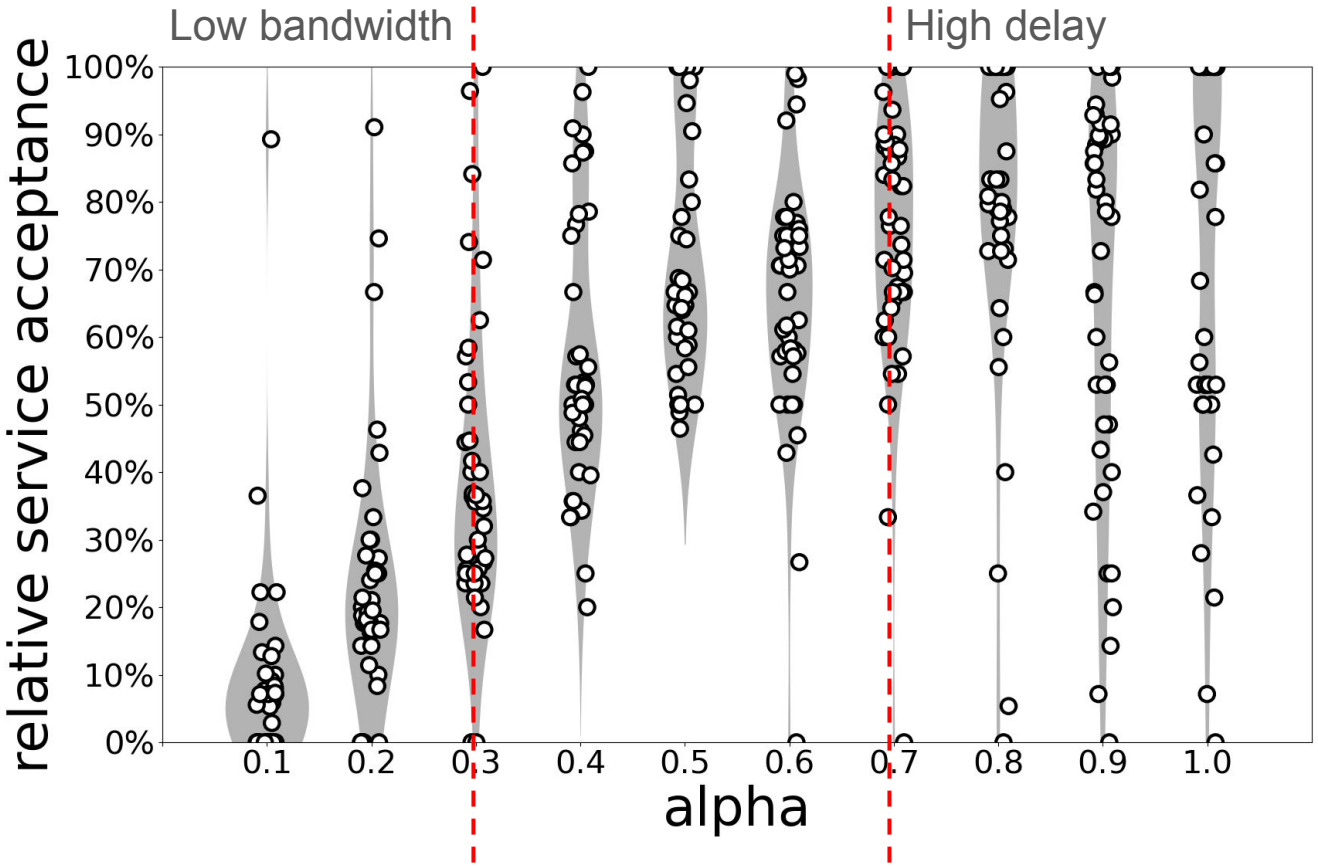
Results

1. General test of quality

The α tradeoff



The α tradeoff



Testing

2. Impact of the requirements

Instance creation

Testing instance: nobel-germany with added variance on delay and capacity metrics

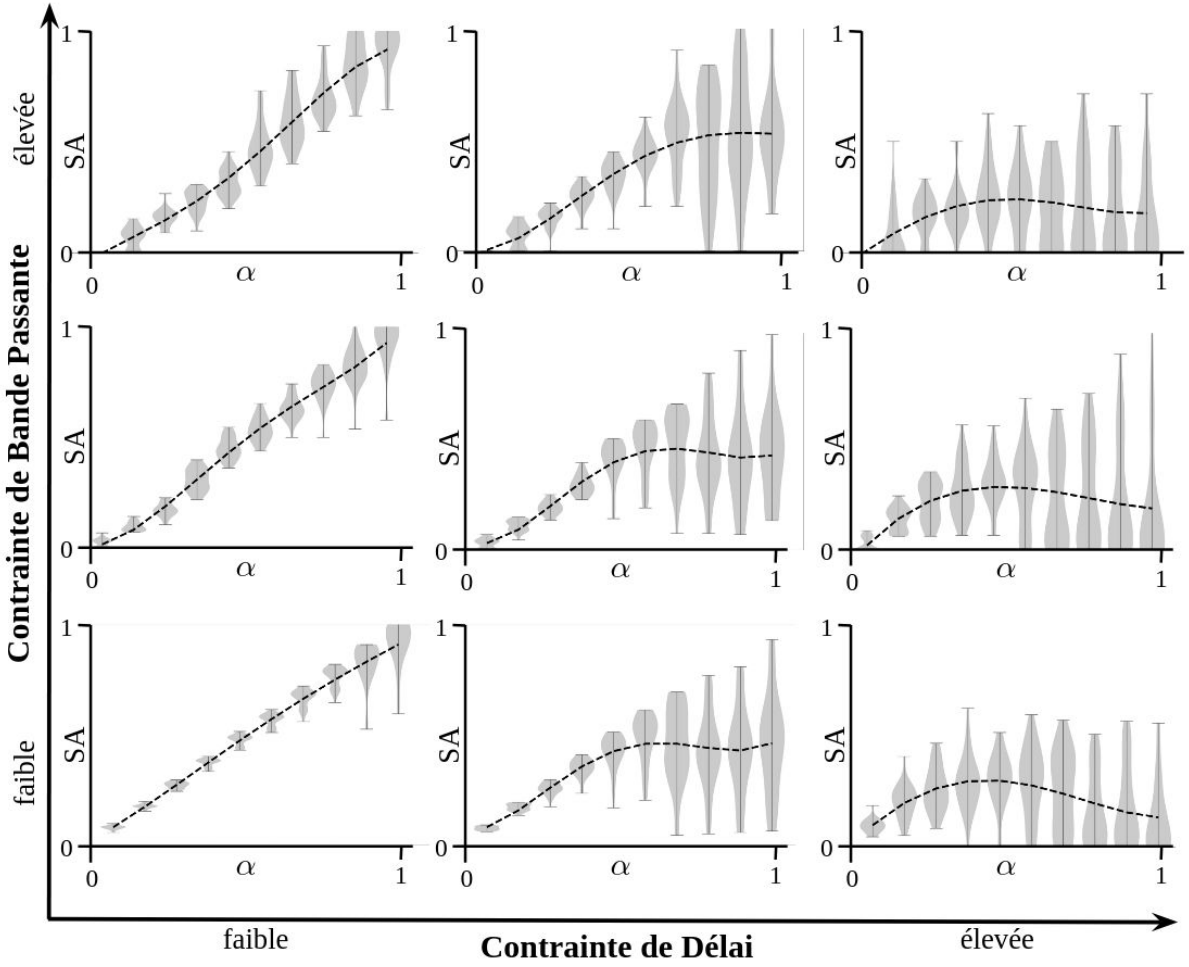
Create 1 family of service requests per virtual link

- Routing problem (1 source, 1 destination; no node requirements)
- 3 **fixed** levels of bandwidth and delay requirements (low - medium - high)
- 50 requests per family

Solving done with the model from Taghavian et al.

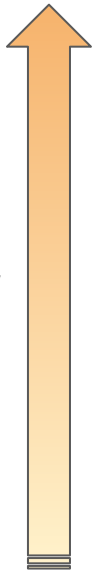
*‘An Approach to Network Service Placement Reconciling Optimality and Scalability’. *IEEE Transactions on Network and Service Management* 20, no. 3 (2023): 2218–29.

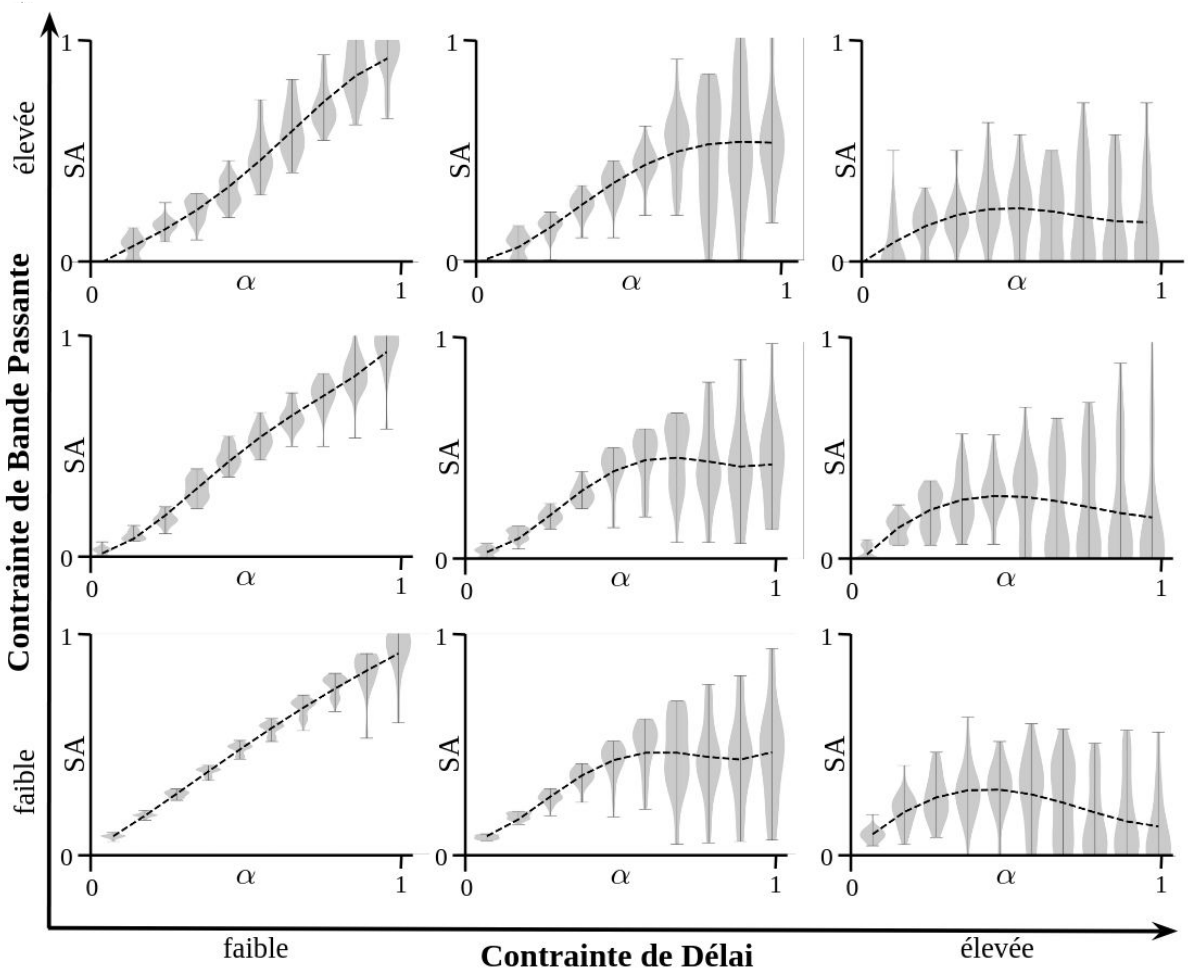
Result


Contrainte de Bande Passante
élevée
faible
SA
α
0
1
Contrainte de Délai
faible
élevée

Result

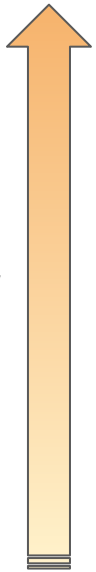
Bandwidth strain has little impact on the solution quality

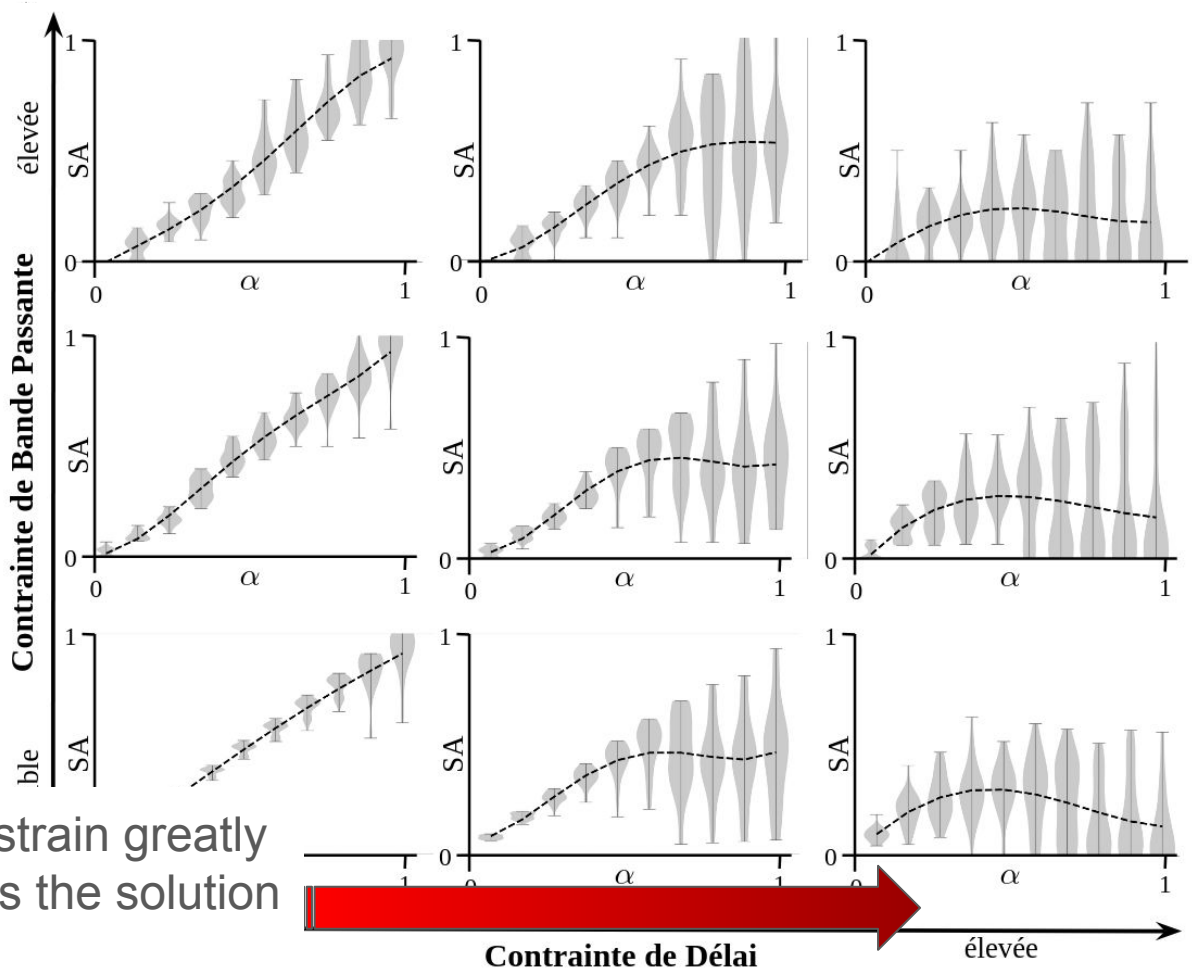




Result

Bandwidth strain has little impact on the solution quality





Delay strain greatly impacts the solution quality

Contents

Perspectives

Current Work

- Implementing a Column Generation model for flow-delay (done in theory, todo in practice)

Future Work

- Test abstraction in online setting (using another tool by Taghavian)
- Relax the under-provisioning condition
- Compare to different abstraction methods (non-legacy)

Thanks for your attention