Fault-Tolerant Application Placement in Heterogeneous Cloud Environments

Bart Spinnewyn, prof. Steven Latré

iMinds
MOSAIC
Modeling Of Systems And Internet Communication
University of Antwerp

Cloud Application Placement Problem (CAPP)

Application Placement

- **admission control**: decide on *acceptance* of application request
- **actual placement**: decide on *how/where* to place application on physical infrastructure

State-of-the-art in cloud management

Typical data center environment

- centralized
- Powerful, homogeneous servers
- wired infrastructure
- ***reliable*** nodes and links

IoT needs computational model

IoT

- Significant computational Aspect

Cloud

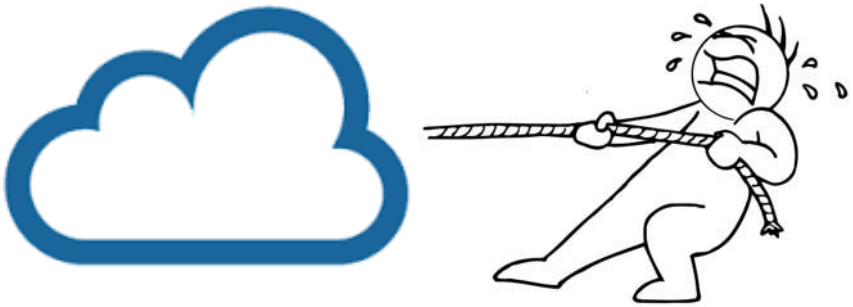
- Provides resources on-demand

However:
Excessive **latency** between local network and cloud infrastructure

Research question

- *How to provide IoT applications with their **computational need**s, while at the same time meeting their stringent **timing constraints**?*

Answer: By extending the cloud into the local network



- use infrastructure that is already available
- minimal latency

Challenge: Availability

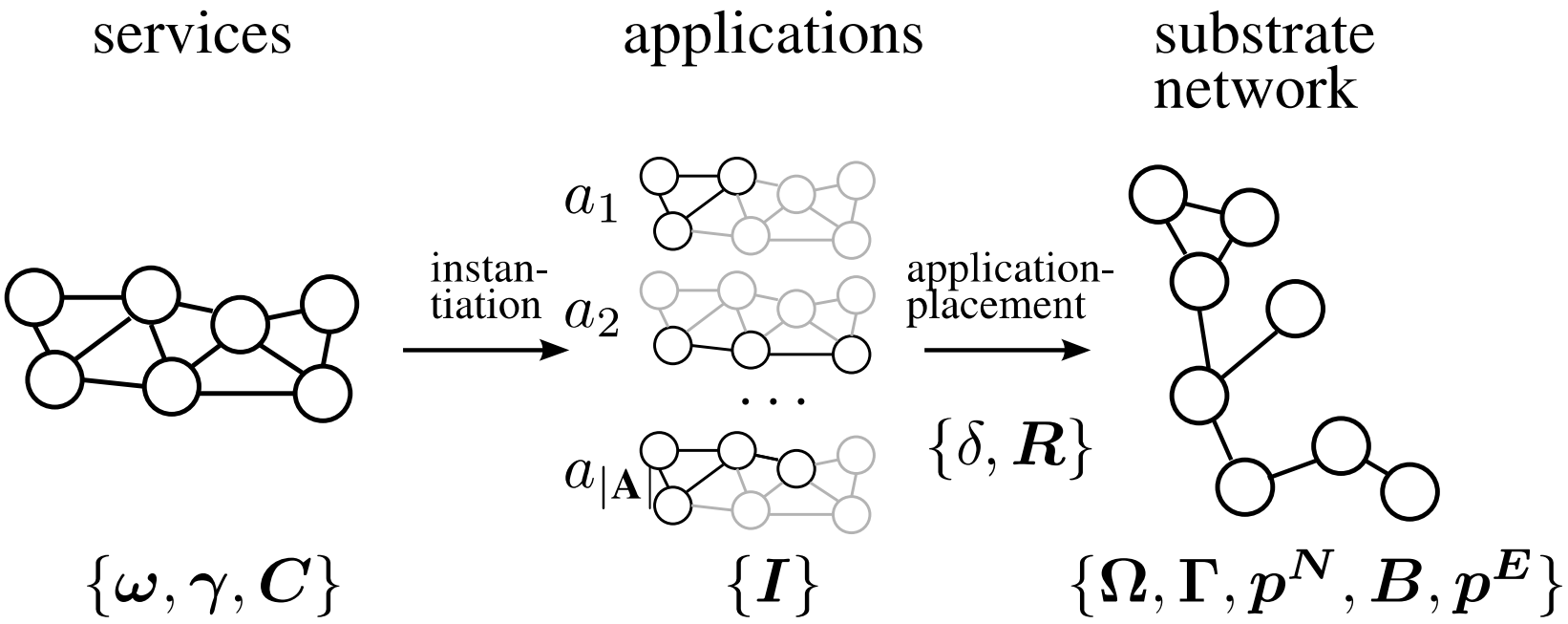
Availability= fraction of time during which application is accessible

But: Unreliable *nodes* and *links in substrate network*

Q: How to guarantee level of *availability* for applications?

A: Fault-tolerant application placement

Formal problem description



Availability-aware application placement

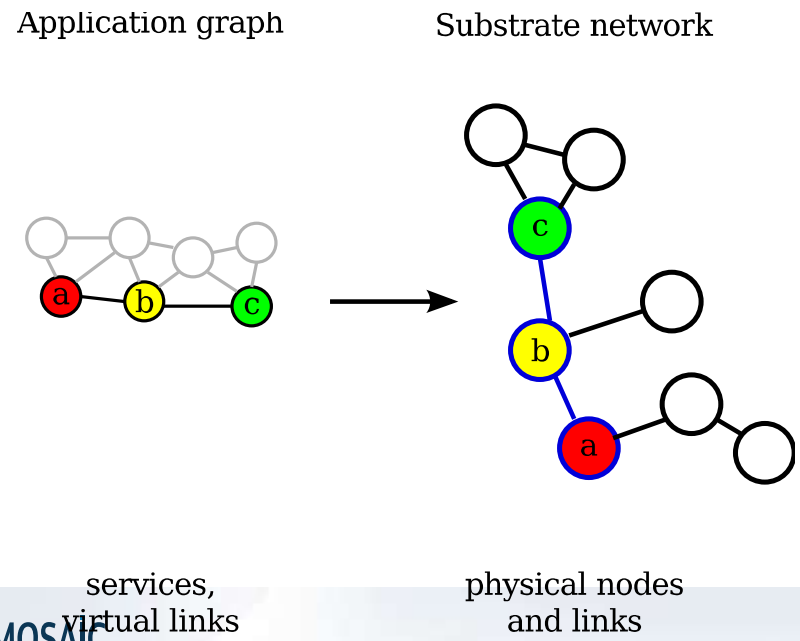
- ***Irredundant***: each service and virtual link is placed at most once
- ***Redundant***: improved availability by multi-path and node replication

When is an application available?
Irredundant placement

For each application:

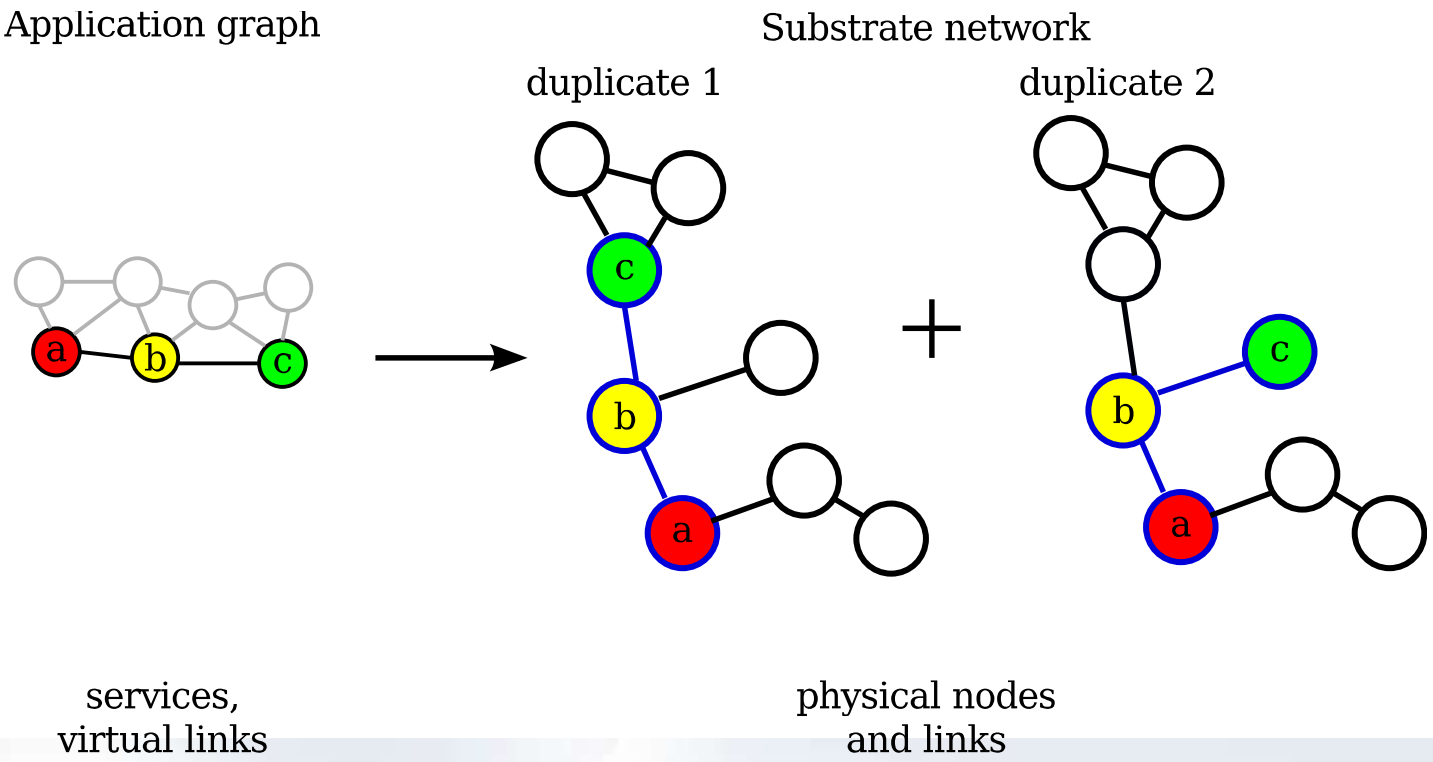
- Place each service and virtual link at most once
- Generate list of physical components (nodes and links) used

=> Application is available when none of the components used fail

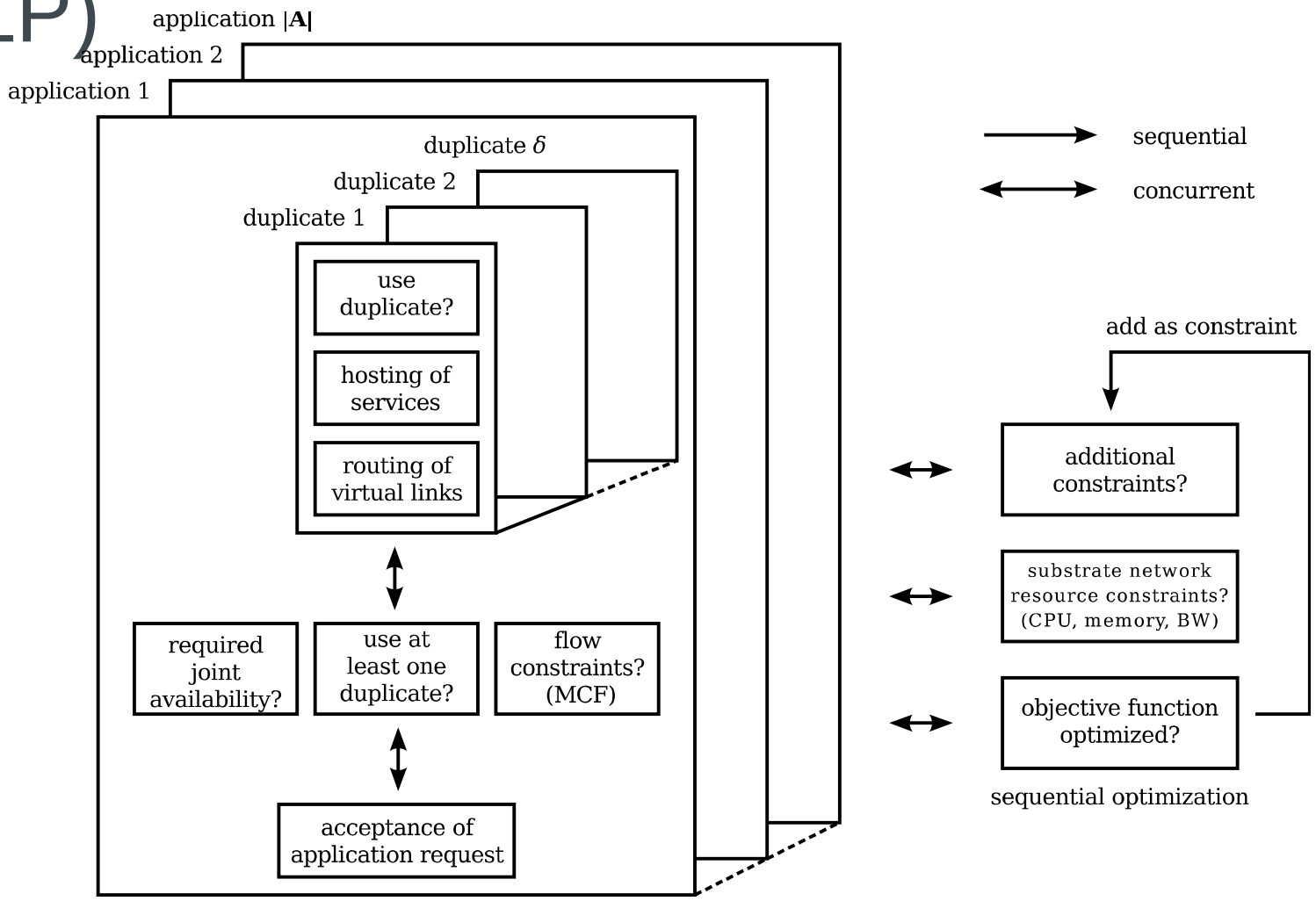


When is an application available?
Redundant placement

- Introduce duplicates: a complete placement of the application (all services and virtual links placed)

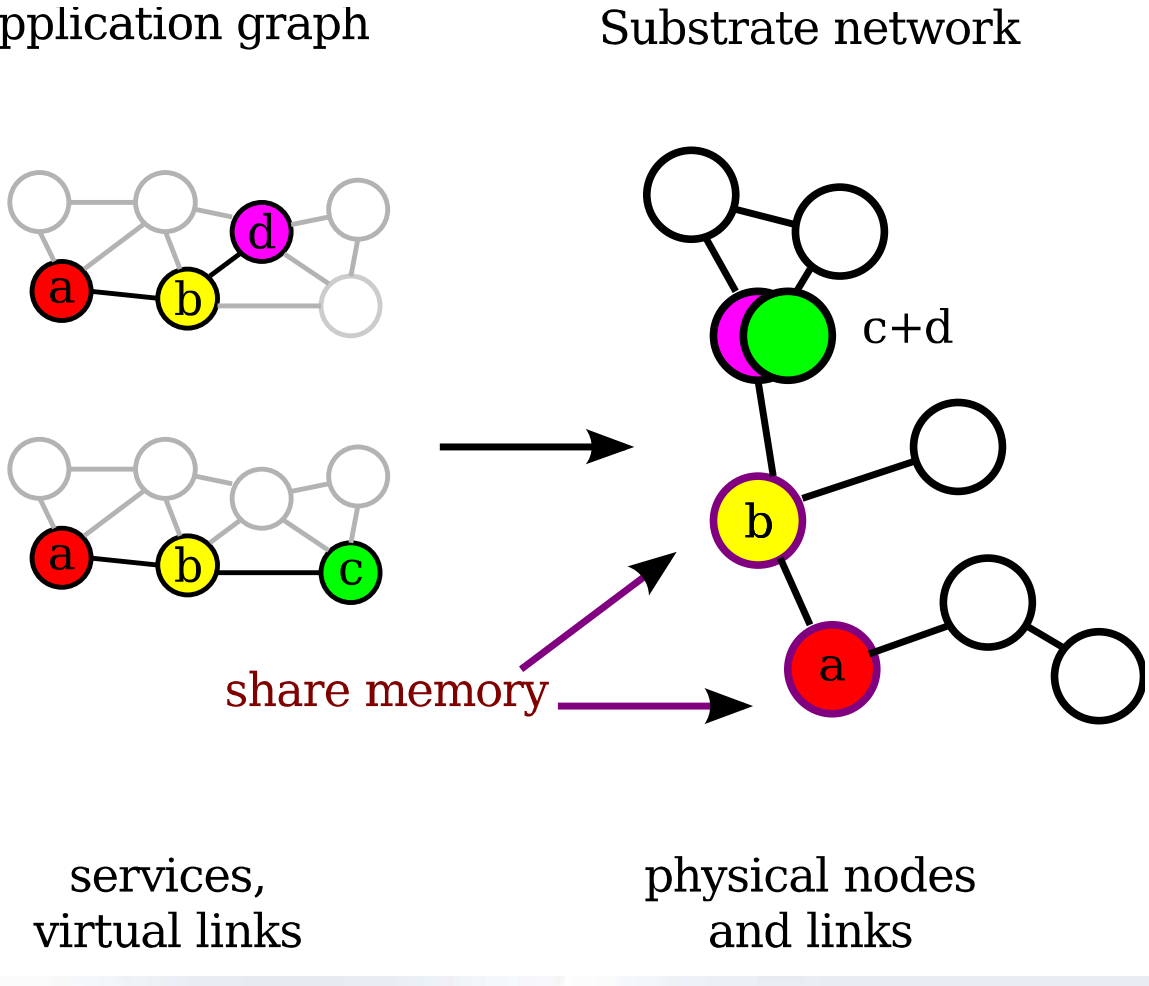


Overview of Integer Linear Program (ILP)



Sharing of resources

- Duplicates are part of multiple applications

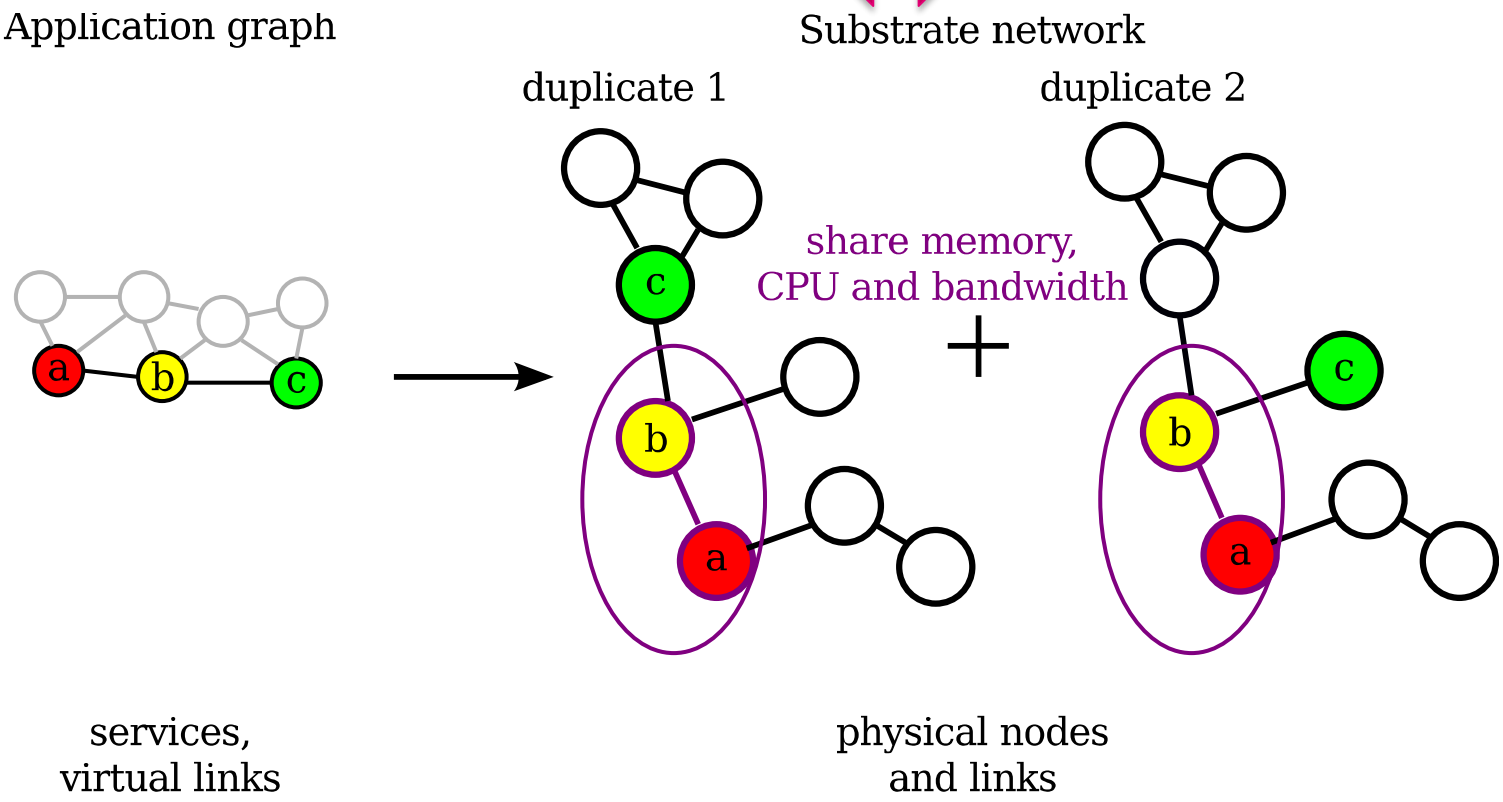


Sharing of resources

- Duplicates are part of the same application

⇒ Only 1 duplicate active at the same time

- **Trade-off**: re-use of resources ⟺ increased availability



Constraints: Availability calculation (1)

Does duplicate rely on component?

- Service
- virtual link

A duplicate is available if none of the components used fail

$$\forall a \in \mathbf{A}, d \in \mathbf{D} : \zeta(a,d) = \mathrm{P}\left[\bigcap_{c \in \mathbf{C}} (\chi_c = 1) \cup (K_c^{d,a} = 0)\right]$$

↓ Rule of Bayes

$$= \sum_{m \in \mathbf{M}} \tau_m^{d,a} \mathrm{P}\left[\mathbf{X} = \mathbf{X}(m)\right],$$

Is the duplicate available, given a particular state?

Expand for all possible states of the substrate network

Constraints: Availability calculation (2)

- Is at least one of the duplicates available?

$$\forall m \in \mathbf{M}, a \in \mathbf{A} : T_m^a \leq \sum_{d \in \mathbf{D}} \tau_m^{d,a}.$$

- Is the availability requirement met?

$$\forall a \in \mathbf{A} : 1 - O^a + \sum_{m \in \mathbf{M}} T_m^a \mathbf{P}\left[\mathbf{X}{=}\mathbf{X}(m)\right] \geq R_a.$$

Can request be Accepted?

Joint availability

Required availability

Objective function

Sequential multi-objective optimization

1. Maximize **acceptance**

$$f_1(\mathbf{A}) = -\sum_{a \in \mathbf{A}} O^a.$$

1. Minimize **BW** usage

$$f_2(\mathbf{A}, \mathbf{E}, \mathbf{S}, \boldsymbol{\beta}) = \sum_{a \in \mathbf{A}} \sum_{e \in \mathbf{E}} \sum_{s_1, s_2 \in \mathbf{S}} \Upsilon^a_{s_1, s_2}(e) \times \beta_{s_1, s_2}.$$

1. Minimize **CPU** usage

$$f_3(\mathbf{A}, \mathbf{N}, \mathbf{S}, \boldsymbol{\omega}) = \sum_{n \in \mathbf{N}} \sum_{a \in \mathbf{A}} \sum_{s \in \mathbf{S}} \Pi^a_{s,n} \times \omega_s.$$

1. Minimize **duplicates** usage

$$f_4(\mathbf{A}, \mathbf{D}) = \sum_{a \in \mathbf{A}} \sum_{d \in \mathbf{D}} G^{d,a}.$$

Performance evaluation:

- CPU Load Factor (CLF): Measure for CPU load on infrastructure

$$\mathrm{CLF} = \frac{\sum_{s \in \mathbf{S}} \sum_{a \in \mathbf{A}} I_{a,s} \times \omega_s}{\sum_{n \in \mathbf{N}} \Omega_n}$$

← Minimal CPU demand

← Total CPU capacity SN

- Placement ratio: fraction of applications for which availability requirement is met

Performance evaluation: algorithms

Algorithm	Author	δ	Availability-aware	Redundancy considered
1	this work	1	yes	no
2	this work	2	yes	yes
3	this work	3	yes	yes
4	Moens et al. [9]	-	no	no

Evaluation Setup

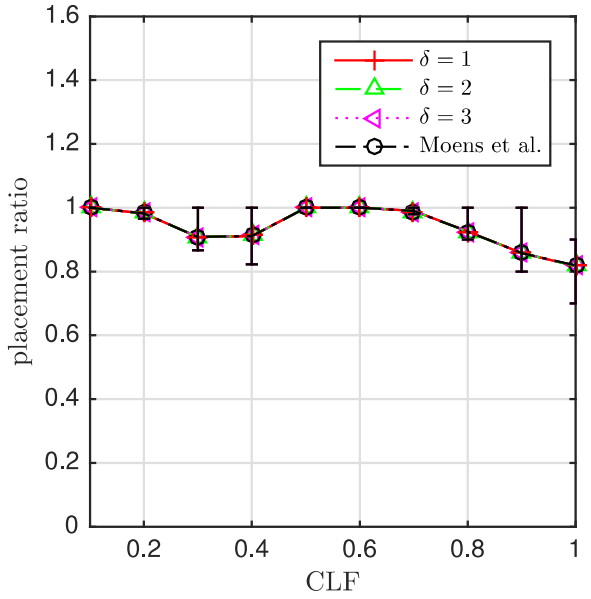
Application graph:

- 10 applications
- 3 services

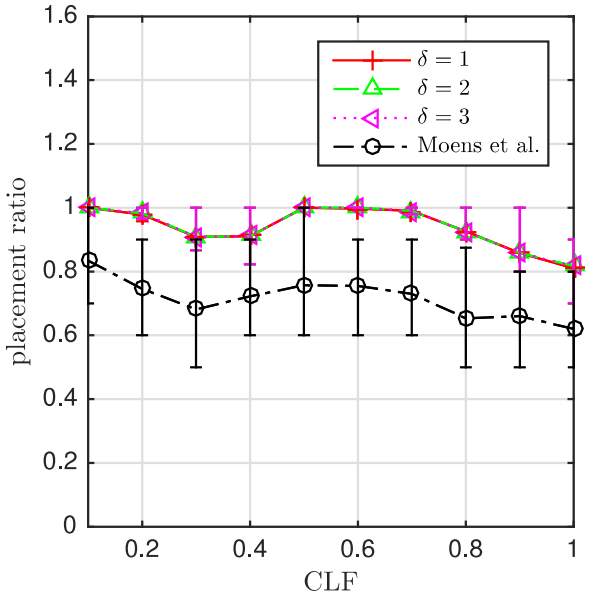
Network graph:

- 5 physical nodes
- 8 edges
- Component failure $\in \{0\%, 2.5\%, 5\%\}$

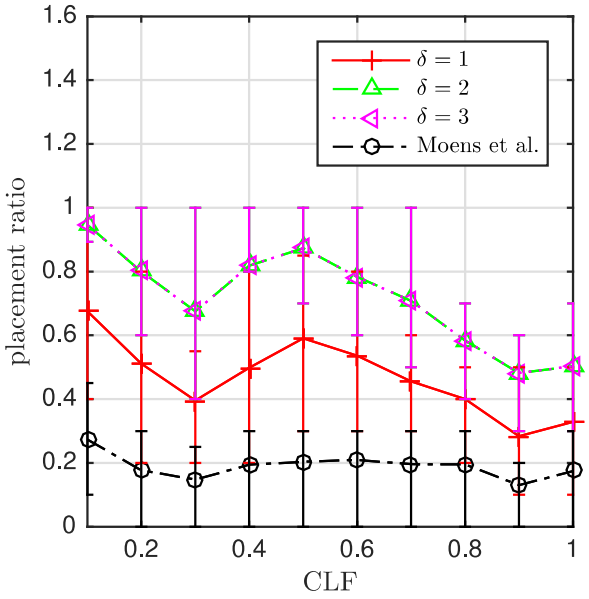
Results: Placement ratio



0%



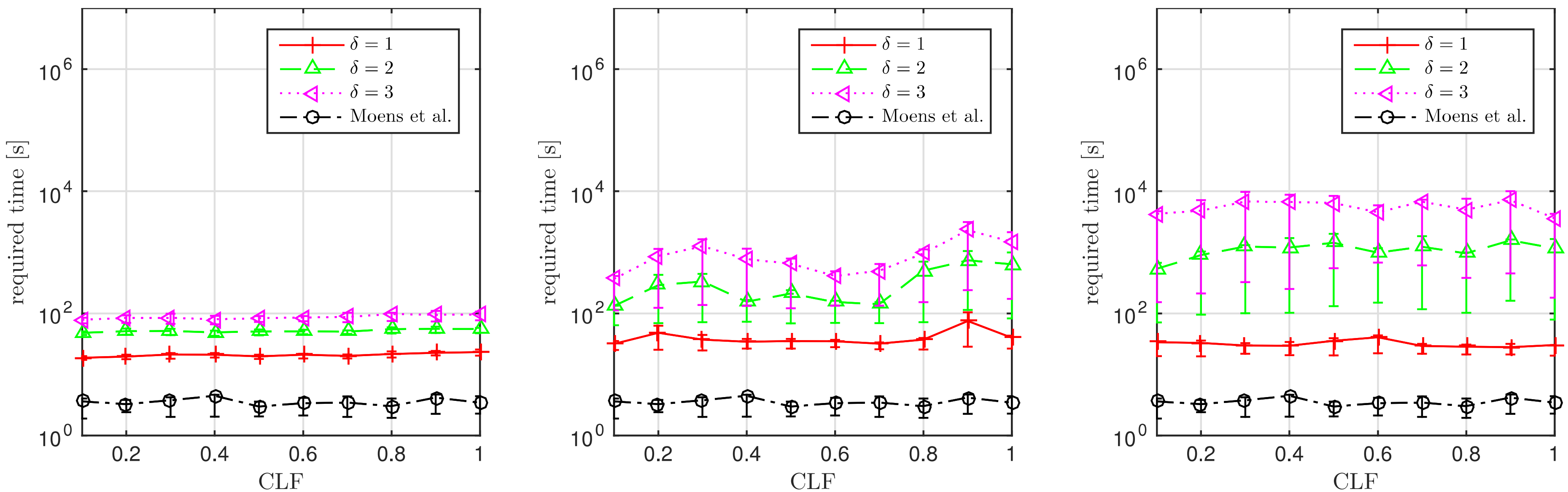
90%



99%

Required availability level

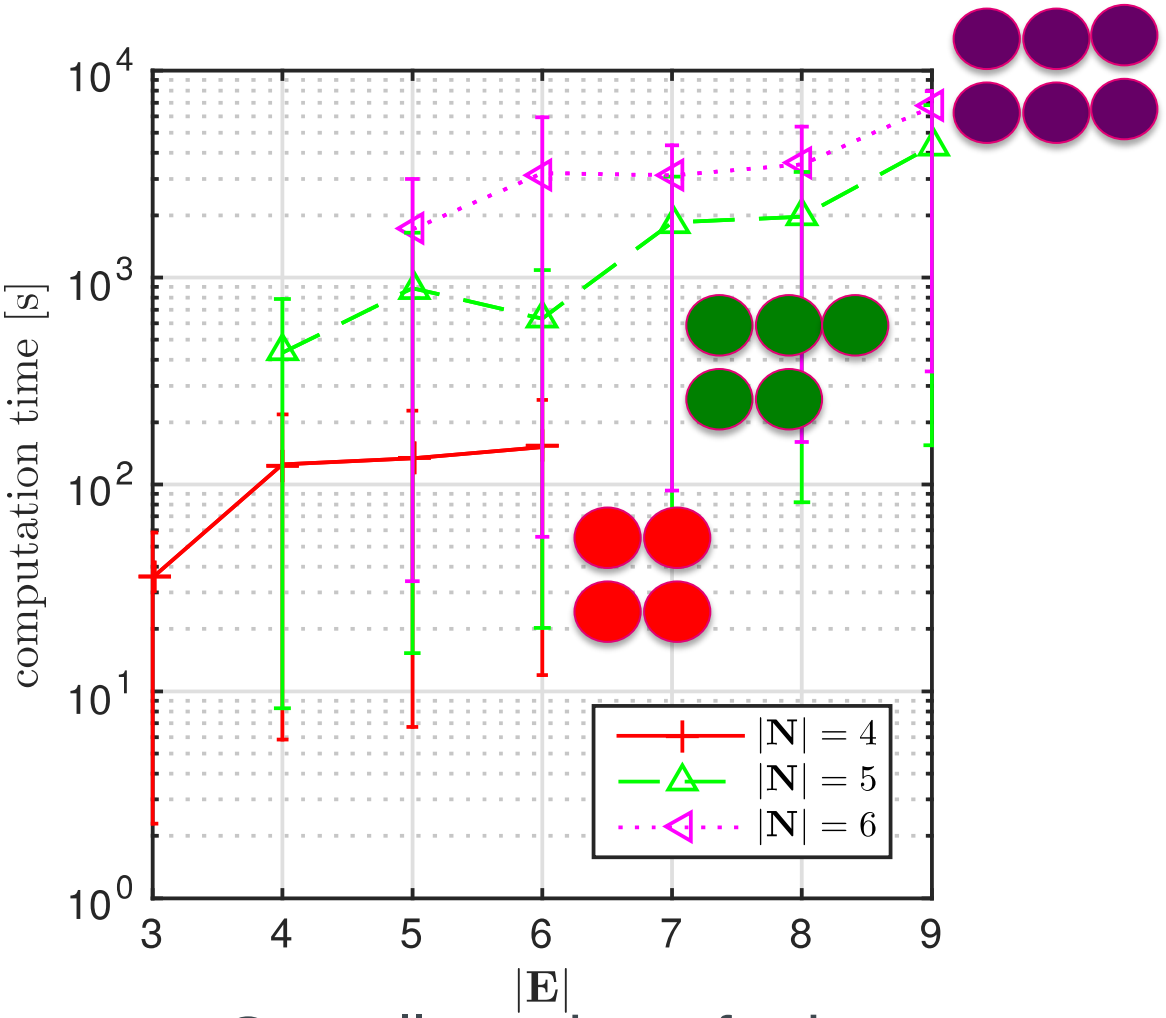
Results: Computation time



0% 90% 99%

Required availability level

Computation time as substrate dimensions vary



Overall number of edges

Conclusion

- Demonstrate need for availability-aware application placement
- Redundancy improves placement ratio

However:

- ILP scales badly for increasing complexity

⇒ need for heuristics

- Dynamic migration
- Static failure model

Questions?

References

[1] B. Spinnewyn and S. Latre, “Towards a fluid cloud: an extension of the cloud into the local network,” in AIMS 2015 PhD workshop, Ghent, Belgium, jun 2015

[9] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck, “Hierarchical network-aware placement of service oriented applications in clouds,” in *2014 IEEE/IFIP Network Operations and Management Symposium, Proceedings*, 2014, pp. 1–8