Modeling and optimization of 5G network design

Wesley da Silva Coelho

Centre d'études et de recherche en informatique et communications
Conservatoire National Arts Métiers
École Doctorale Sciences des Métiers de l'Ingénieur

Bernardetta ADDIS	Université de Lorraine	Rapporteuse
Pierre FOUILHOUX	Paris Nord	Rapporteur
Safia KEDAD-SIDHOUM	Conservatoire National des Arts et Métiers	Examinatrice
Tijani CHAHED	Telecom SudParis	Examinateur
Bernard FORTZ	Université Libre de Bruxelles	Examinateur
Stefano SECCI	Conservatoire National des Arts et Métiers	Directeur de thèse
Amal BENHAMICHE	Orange Labs	Co-encadrante
Nancy PERROT	Orange Labs	Co-encadrante

Sciences des Métiers de l'Ingénieur

HESAM UNIVERSITÉ

le cnam

AGENDA

1 - Introduction

- Contextualization
- Modeling Aspects

2 - The Network Slice Design Problem

- Problem Definition
- Complexity
- Mathematical Formulation
- Sensibility Analyses

3 - Exact Approaches

- Branch-and-Cut
- Row Generation

4 - Heuristic Approaches

- Math-Heusristic
- Relax-and-Fix

5 - Concluding Remarks

- Summary
- Perspectives

CONTEXTUALIZATION

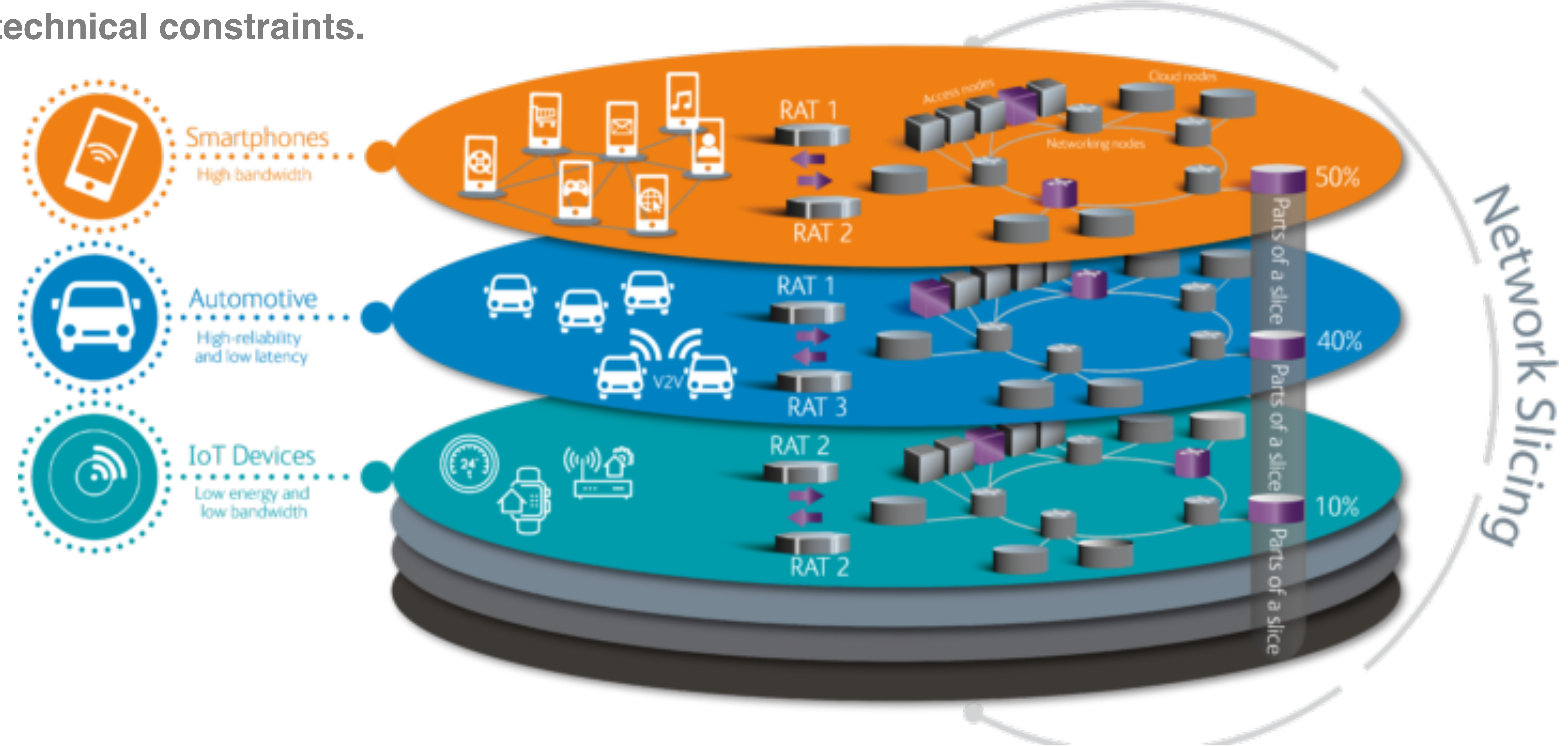
Where we are? Where we are going?

The age of boundless connectivity

- Different types of services
  - autonomous car
  - virtual reality
  - industry 4.0
  - …
- New needs
  - speed
  - capacity
  - availability
  - …
- A solution
  - Network Slicing

What is Network Slicing?

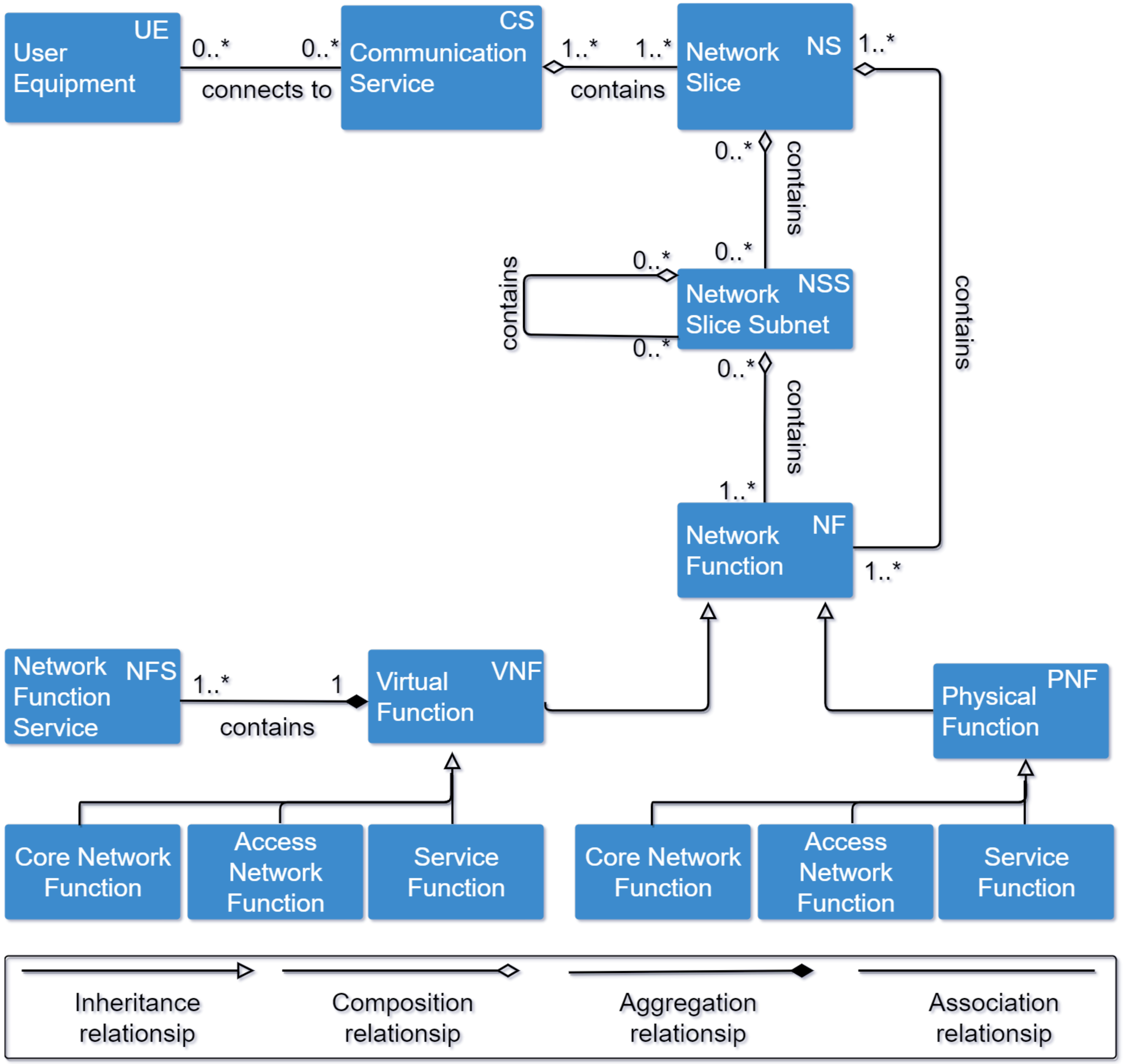
- Set of **logical networks** on top of a **shared physical infrastructure**.
  - Each logical network is designed to serve a **defined business purpose**.
  - Comprises of all the required **network resources.**
  - Ensures all **technical constraints.**



Modeling Aspects

How to deploy an end-to-end network slice.

The 5G entities



User Equipment (UE)

- mobile phone, automous car, robot …

Communication Service (CS)

- video streaming, industry 4.0, IoT…

Network Slice (NS)

- service-tailored virtual network

Network Slice Subnet (NSS)

- control-plane NSS, data-plane NSS, …

Virtual Network Function (VNF)

- SMF, AMF, UPF, Load Balancer, Proxy, …

Network Function Service (NFS)

- consumer/provider-based micro-services

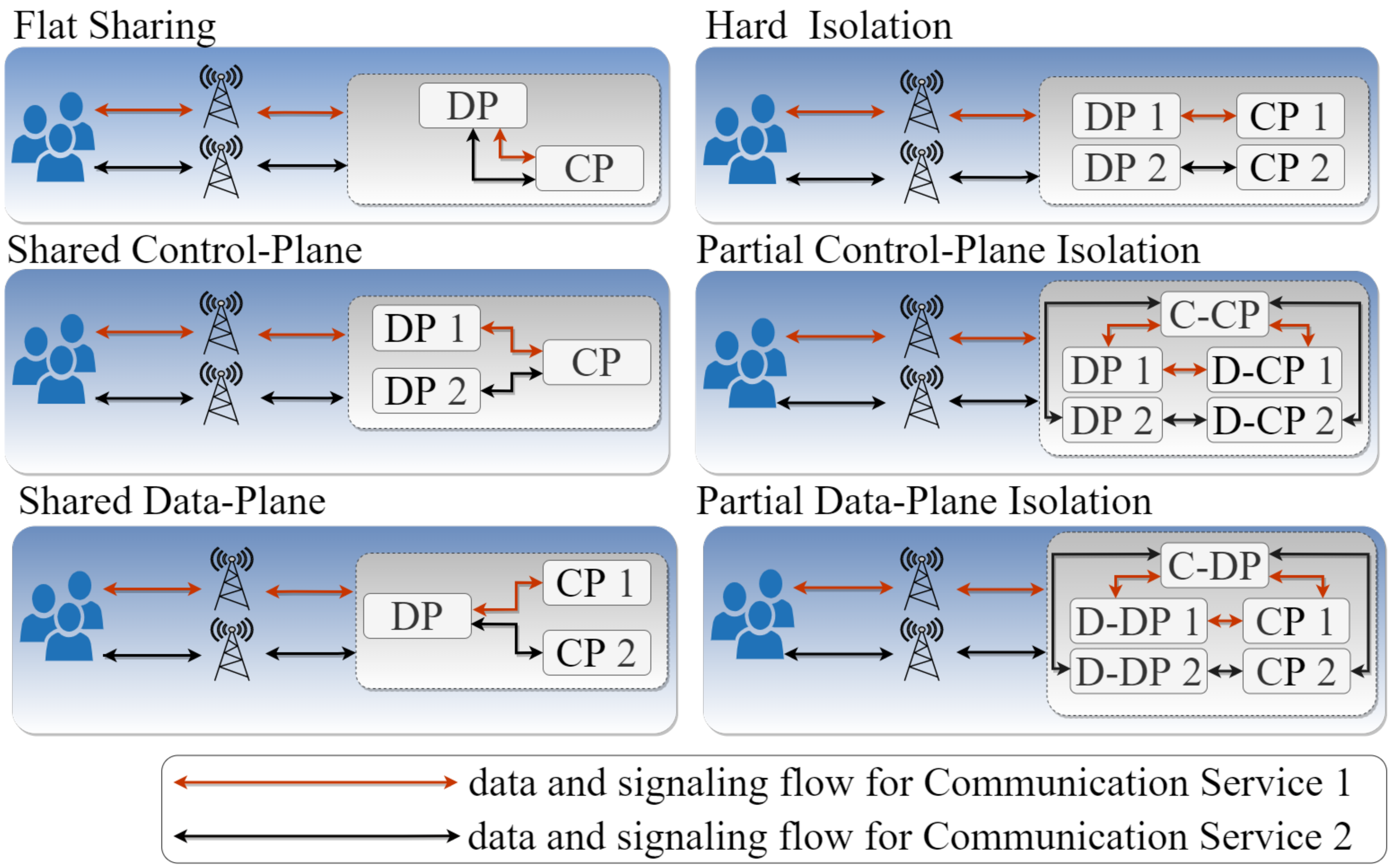
Control and data plane separation and sharing policies

Isolation

- Security requirements
- Easier network management

Sharing

- Decrease redundancy
- Faster set-up



Control and data plane separation and sharing policies

Flat Sharing
- Similar technical constraints

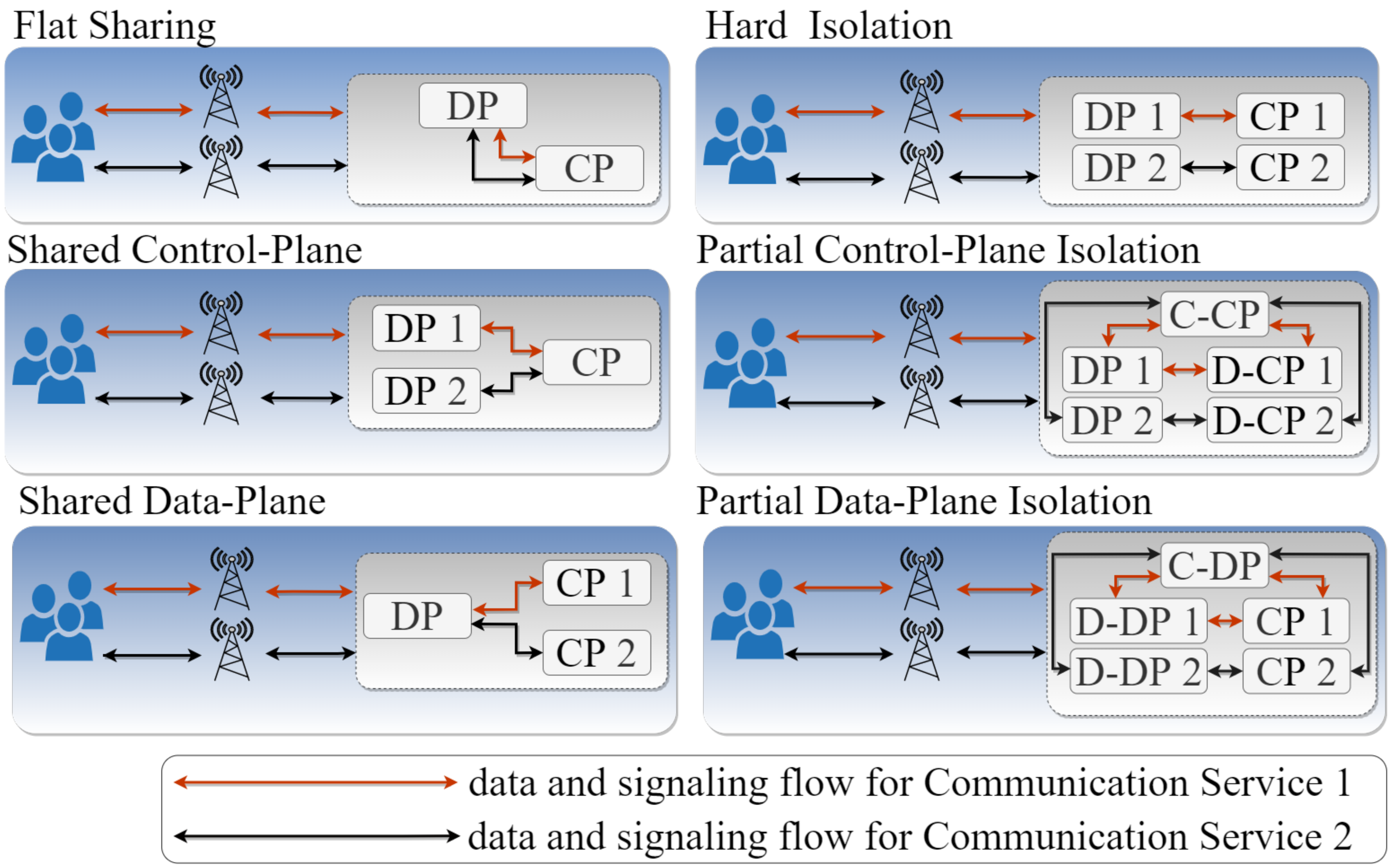
Hard Isolation
- Security : slices are fully isolated

Shared Control-Plane
- Low-latency on dedicated Data-Plane (D-DP)

Partial Control-Plane Isolation
- Sharing only non-crucial CP functions

Shared Data-Plane
- Dedicated Control-Plane (D-CP)

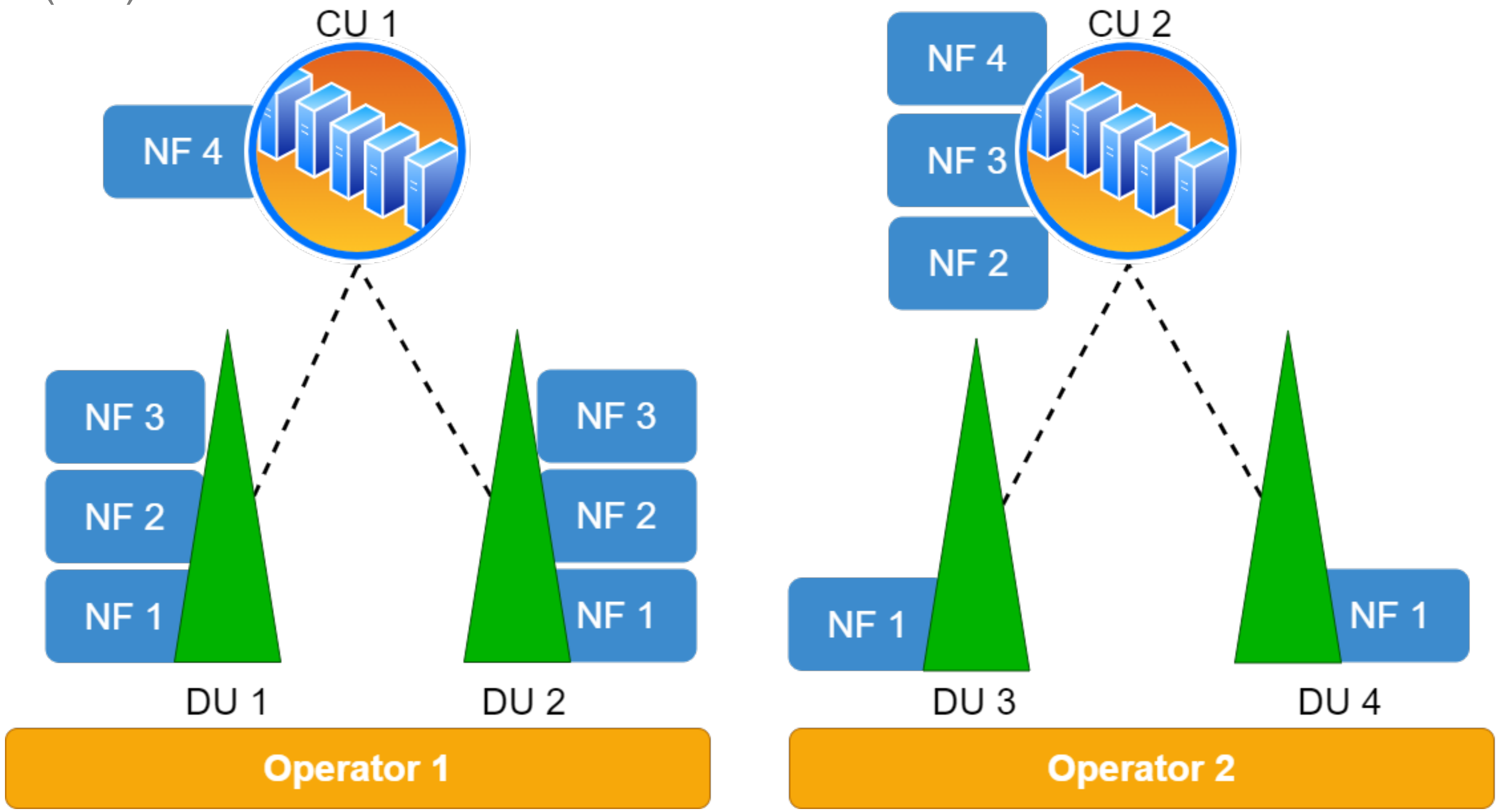
Partial Data-Plane Isolation
- Sharing only non-crucial DP functions



Functional splitting on radio access networks

- Decide which networks functions (NF) are:
  - Locally installed on each distributed unit (DU)
  - Centrally installed on few centralized units (CU)

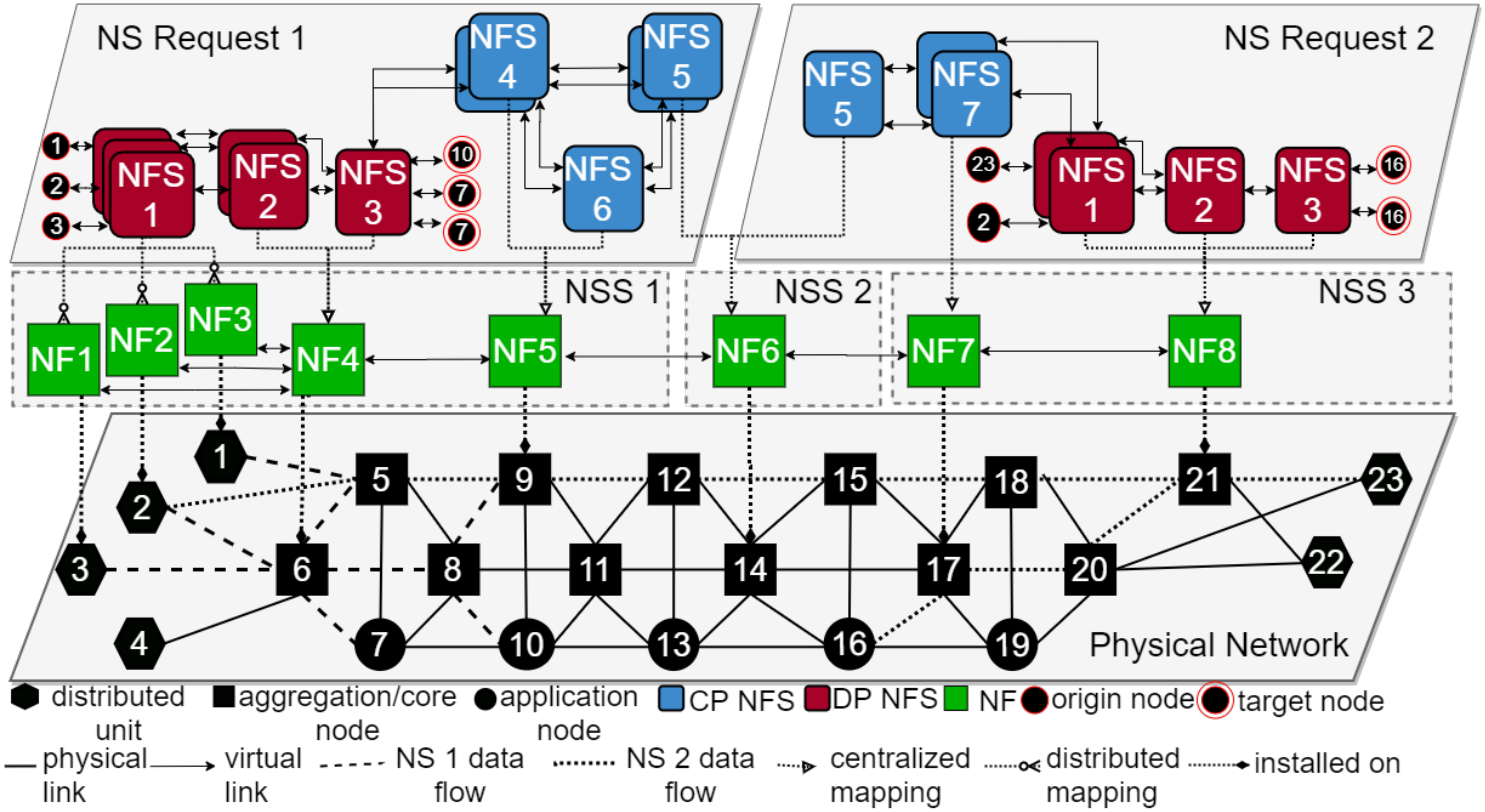
- Must take into consideration
  - Fronthal bandwidth
  - DU-CU connection latency



The Network Slice Design Problem

Definition :: Complexity :: Variants :: MILP :: Sensibility Analysis

Network Slice Design Problem



Problem Statement

Given:

- a directed graph *G* representing the **physical network**,
- a set of **slice requests** *S*
    - a set of **traffic demands** *K(s)* associated with each request *s* in *S*
    - and a set *F* of **NFS types**

The **Network Slice Design Problem** (NSDP) consists in **determining:**

- the **number of *NFSs*** to install for each slice,
- the **size of each *NF*** hosting them
    - whether they are to be installed **centrally or distributed**

So that:

- *NFSs* installed on *G* must be **packed** into the *NFs* while satisfying both **isolation** and **capacity** constraints
- the data-flow between any pair of NFs can be **controlled and routed** in *G*
- the **deployment cost is minimized**

The Network Slice Design problem is <u>**NP-complete**</u> even with only one slice request.

A Mixed-Integer Linear Programming Formulation for the NSDP

$$min \sum_{f\in F}\sum_{n\in N}\sum_{u\in V_p} y^f_{nu} + \Omega \sum_{a\in A_p}\sum_{s\in S}\sum_{k\in K(s)}\sum_{f,g\in F} \gamma^{ka}_{fg}$$

$$z^s_f \leq z^s_{f+1}, \qquad \forall s \in S, \forall f \in F^d \backslash \{f_{|F^d|}\}.$$

$$cap(f) w^{sf}_{nu} = \begin{cases} \sum\limits_{k\in K(s)|u=o_k} \lambda_{f-1} b^k x^{sf}_{nu} & \text{if } f \in F^d, u \in V^{du} \\ n_s b_f x^{sf}_{nu} & \text{if } f \in F^c; \\ \sum\limits_{k\in K(s)} \lambda_{f-1} b_k x^{sf}_{nu} & \text{if } f \in F^d, u \in V^{ac}. \end{cases} \qquad \forall s \in S, \forall f \in F, \forall n \in N, \forall u \in V$$

$$\sum_{s\in S} w^{sf}_{nu} \leq y^f_{nu} \qquad \forall n \in N, \forall v \in V, \forall f \in F$$

$$x^{sf}_{nu} + x^{tg}_{nu} \leq 1 + q^{st}_{fg} q^{ts}_{gf} \qquad \forall s,t \in S, u \in V, n \in N, f,g \in F$$

$$\sum_{n\in N} x^{sf}_{nu} + \sum_{m\in N} x^{tg}_{mu} \leq 1 + q^{st} q^{ts} \qquad \forall s,t \in S, u \in V^{na}, f,g \in F$$

$$x^{sf}_{nu} + x^{tg}_{nv} \leq 1 \qquad \forall s,t \in S, f,g \in F, n \in N, u,v \in V$$

$$\sum_{n\in N} x^{sf}_{nu} = \begin{cases} 1 - z^s_f & \text{, if } f \in F^d, u = o_k; \\ 0 & \text{, otherwise .} \end{cases} \qquad \forall k \in k(s)|s \in S, \forall f \in F, u \in V^{du}$$

$$\sum_{n\in N}\sum_{u\in V^{ac}} x^{sf}_{nu} = \begin{cases} z^s_f & \text{, if } f \in F^d; \\ \alpha^s_f & \text{, if } f \in F^c. \end{cases} \qquad \forall s \in S, \forall f \in F$$

$$\sum_{a\in\delta^+(u)} \gamma^{ka}_{fg} - \sum_{a\in\delta^-(u)} \gamma^{ka}_{fg} = \begin{cases} z^s_f - 1 & \text{if } (f,g) \in G(s), f \in F^c, u = o_k, \\ 1 - z^s_f & \text{if } , u = o_k, ((f,g) \in G(s), f \in F^d) \oplus (f = f_{|F^d|}, g = f_0); \\ -\sum\limits_{n\in N} x^{sg}_{nu} & \text{if } u \in V\backslash V^{du}, f = f_0, g = f_1 | g \in F^d \\ z^s_g & \text{if } u = o_k, f = f_0, g = f_1 | g \in F^d \\ -1 & \text{if } u = t_k, f = f_{|F^d|}, g = f_0 \\ \sum\limits_{n\in N} x^{sf}_{nu} & \text{if } u \in V\backslash V^{du}, f = f_{|F^d|}, g = f_0 \\ z^s_g - z^s_f & \text{if } u = o_k, \forall f,g \in F^d | g = f+1 \\ \sum\limits_{n\in N} x^{sf}_{nu} - \sum\limits_{m\in N} x^{sg}_{mu} & \text{otherwise.} \end{cases} \qquad \forall k \in K(s) : s \in S, \forall f,g \in F, \forall u \in V,$$

$$\sum_{a\in A} d_a (\gamma^{ka}_{f_{|F^d|} f_0} + \sum_{f\in\{f_0\}\cup F^d\backslash\{f_{|F^d|}\}} \gamma^{ka}_{ff+1}) \leq d_s \qquad \forall k \in K(s) : s \in S$$

$$\sum_{a\in A} d_a \gamma^{ka}_{fg} \leq d_{fg} \qquad \forall k \in K(s) : s \in S, \forall f,g \in F$$

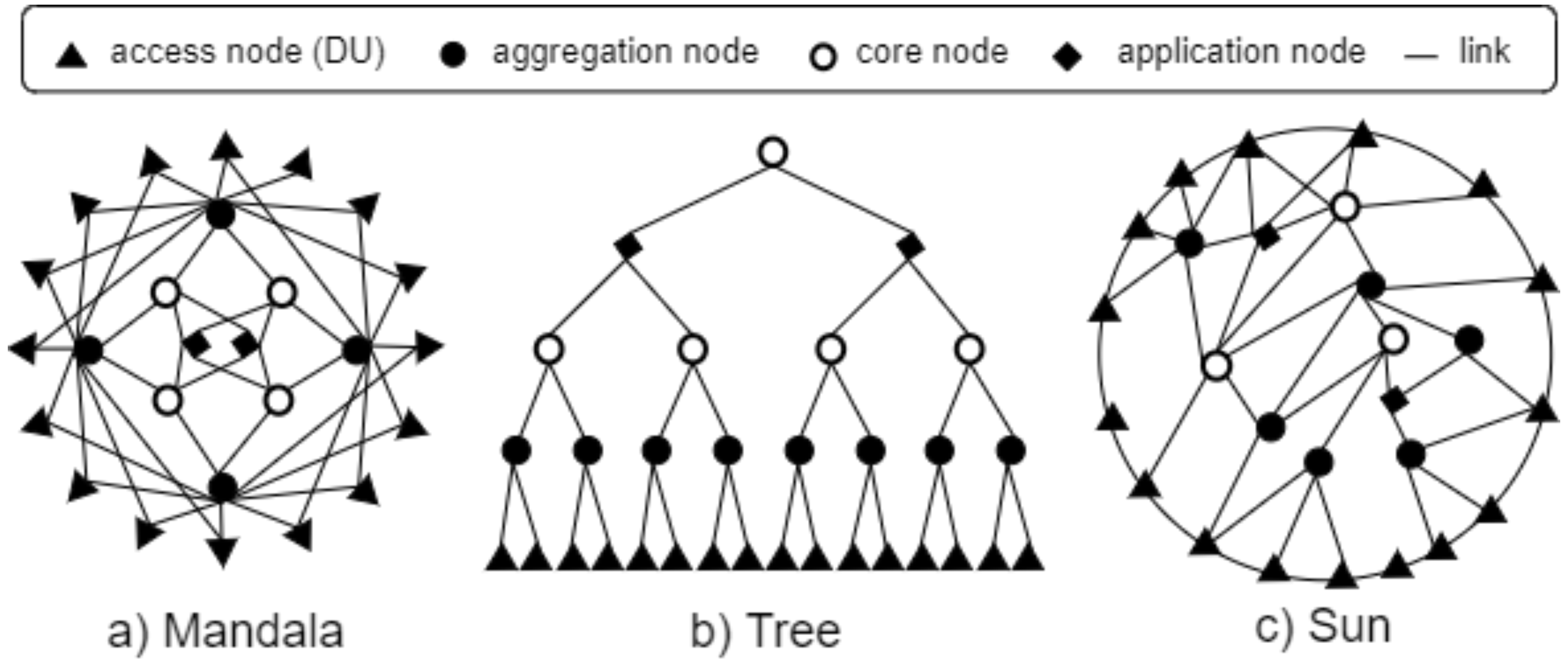
$$\sum_{s\in S}\sum_{k\in K(s)} b^k (\lambda_{f_{|F^d|}} \gamma^{ka}_{f_{|F^d|} f_0} + \sum_{f\in\{f_0\}\cup F^d\backslash\{f_{|F^d|}\}} \lambda_f \gamma^{ka}_{ff+1})] + \sum_{s\in S} n_s (\sum_{(f,g)\in F(s)} b_{fg} \gamma^{1a}_{fg} + \sum_{(f,g)\in G(s)}\sum_{k\in K(s)} \frac{b_{fg}\gamma^{ka}_{fg}}{|K(s)|}) \leq b_a \qquad \forall a \in A$$

$$\sum_{n\in N}\sum_{f\in F} c^c_f y^f_{nu} \leq c^c_u \qquad \forall u \in V, \forall c \in C$$

Sensibility Analysis

Test setting

- Virtual Layer
  - 5 data-plane NFSs
    - NFS1 : functions from MAC bloc
    - NFS2 : functions from RLC bloc
    - NFS3 : functions from PDCP bloc
    - NFS4 : functions from RRC bloc
    - NFS5 : data-plane UPF from 5G core network
  - 8 control-plane NFSs
    - 4 mandatory : NFS6, NFS7, NFS8, NFS9
    - 4 optional : NFS10, NFS11, NFS12, NFS13
  - Processing capacity : 100% the average volume sent by all DUs.
  - CPU requirement : 5% of the average capacity on physical nodes.
  - Compression coefficients for DP NFSs : as calculated in by Larsen et al (2018)
  - Max latency
    - between DP NFSs : as proposed by 3GPP
    - between CP NFSs : 5% of the total CP latency proposed by 3GPP



Sensibility Analysis

Test setting

- Network Slice
  - 4 requests, each of which with 8 traffic demands
  - DP-CP connexion : between one NFS6 and all DP NFSs
  - All CP NFSs must be connected to each other
  - 25% of available DUs are set to be an origin node of all NS requests
  - Target application nodes : evenly distributed

TABLE – Simulated slice demand setting. Source : adapted from NGMN's White Paper (2015)

Slice	Service required	Optional CP NFSs	Max E2E latency	UE data rate	UE per DU
1	Broadband access in dense areas	NFS10, NFS11	10ms	300Mbps	600
2	Ultra-low cost broadband	-	10ms	10Mbps	600
3	Real-time communication	NFS11, NFS12, NFS13	1ms	25Mbps	180
4	Video broadcast	NFS10, NFS11	100ms	200Mbps	60

Sensibility Analysis

TABLE – Scenarios : Split setting and sharing policies

Split	Description
setting 1	all DP NFS are installed locally for all NS requests.
setting 2	for each slice, only NFS5 is installed centrally.
setting 3	for each slice, NFS4 and NFS5 are installed centrally. It correspond to 3GPP's split 1.
setting 4	for each slice, only NFS1 and NFS2 are distributed ; it corresponds to 3GPP's split 2.
setting 5	for each slice, only NFS1 is installed locally. It corresponds to 3GPP's split 4.
setting 6	all DP NFSs are installed centrally for all NS requests. It corresponds to 3GPP's split 6
Flexible	free functional split selection for each NS request.

Policy	Description
Hard Isolation	NS requests do not accept sharing any NFS.
Shared DP	only DP NFSs can be shared among the slices
Shared CP	only CP NFSs can be shared among the slices
Partial DP Isol.	NFS4 and NFS5 cannot be shared among the slices
Partial CP Isol.	optional NFSs cannot be shared among the slices.
Flat Sharing	NS requests do not impose any isolation constraint.

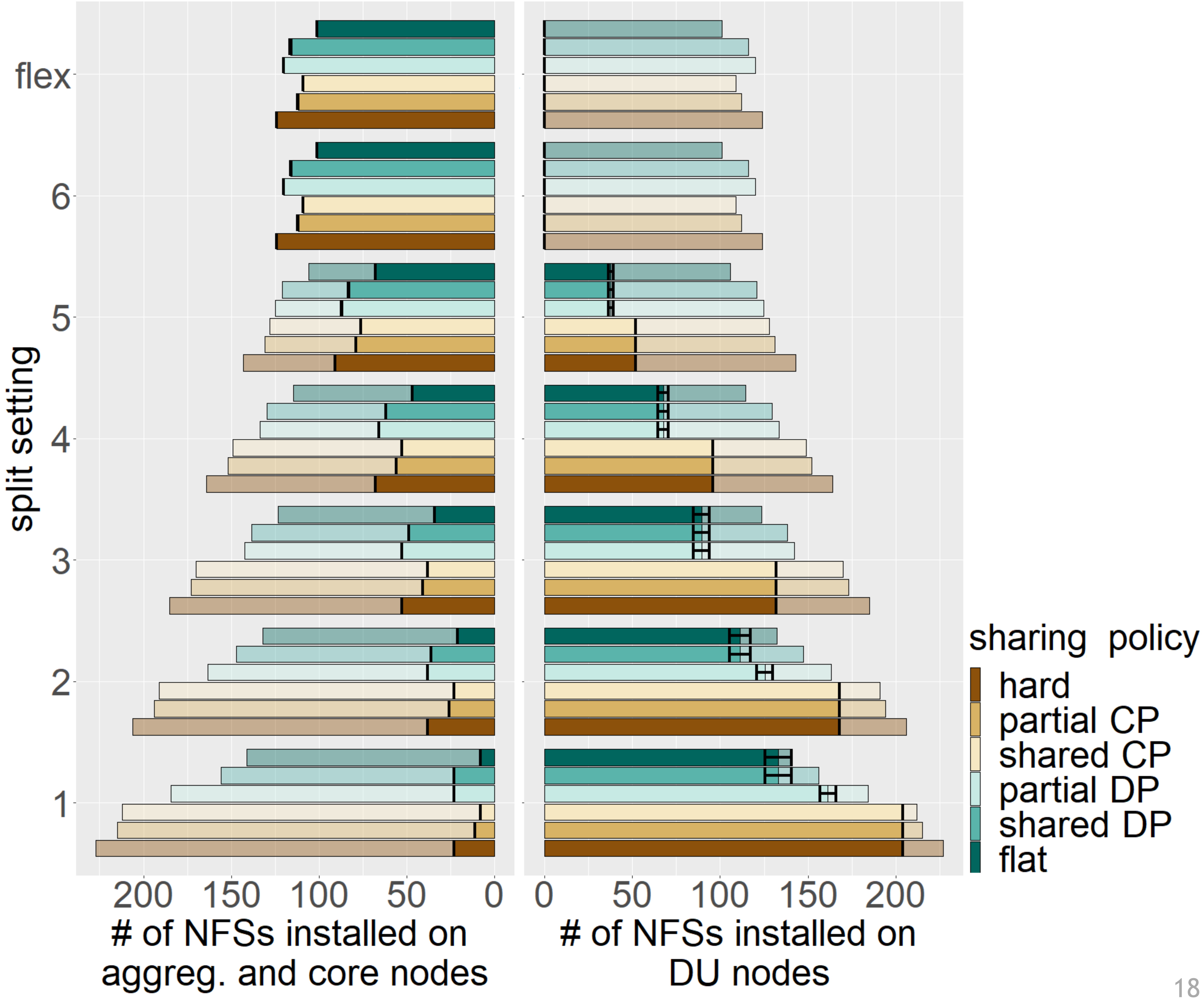
Sensibility Analysis

Numerical Results

Split Setting 1 + Hard Isolation: 227 NFSs installed

Flex Split + Flat Sharing: 101 NFSs installed

Overall reduction : 56%



Sensibility Analysis

Physical node load

- Relatively strong influence from different sharing polices
- Opposite impacts on different physical node types

(a) physical links: usage ratio
(b) physical links: average load
(c) fronthaul links: average load
(d) back./core links: average load
(e) most loaded physical link
(f) number of nodes hosting NFs
(g) physical nodes: average load
(h) DU nodes: average load
(i) aggreg./core nodes: average load
(j) most loaded physical node

Sensibility Analysis

Physical link load

- Strongly impacted by centralized NFS-based split settings
- Relatively small influence from different sharing polices

Exact Approaches for the NSDP

Model Strengthening :: Row Generation

Model Strengthening

Symmetry-breaking constraints

- Assign the NFs in an ordered way
- NF **n** cannot host any NFS if NF **n-1** hosts no NFS

$$x_{nu}^{sf} \leq \sum_{t \in S} \sum_{g \in F} \sum_{v \in V} x_{n-1v}^{tg} \qquad \forall s \in S, \forall f \in F, \forall u \in V, \forall n \in N \backslash \{n_1\}$$

Lower-bound inequality

- minimum number of NFSs needed to satisfy all the slice requests of S

$$\sum_{f \in F^c} \left\lceil \sum_{s \in S} \frac{n_s b_f}{cap(f)} \right\rceil + \sum_{f \in F^d} \left\lceil \sum_{k \in K(s): s \in S} \frac{\lambda_{f-1} b^k}{cap(f)} \right\rceil \leq \sum_{f \in F} \sum_{n \in N} \sum_{v \in V} y_{nu}^{f}$$

Model Strengthening

Shortest path-based inequalities

- For each traffic demand k
  - sp(k) be the end-to-end latency on the shortest path between its origin and target nodes
  - Considering the capacity of the related links
  - Using Dijksitra’s algorithm

$$\sum_{a\in A_p} d_a\Big(\gamma^{ka}_{f_{|F^d|}f_0} + \sum_{f\in\{f_0\}\cup F^d\setminus\{f_{|F^d|}\}} \gamma^{ka}_{ff+1}\Big) \geq sp(k) \qquad ,\forall k\in K(s) : s\in S$$

Example : 100Gbs as expected flow

(a) Without SP valid inequality.

(b) With SP valid inequality.

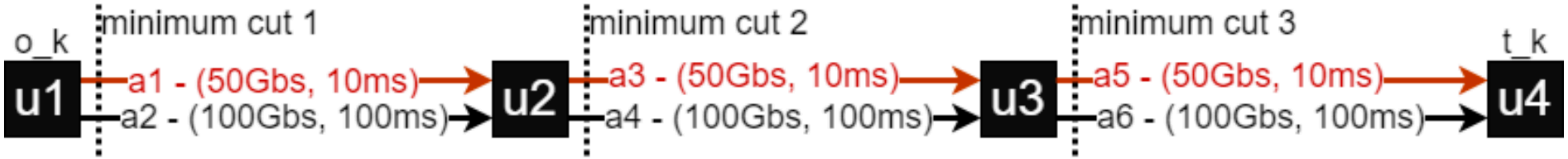
Model Strengthening

Minimum cut-based inequalities

- based on one min-cut max-flow theorem of Ford and Fulkerson
  - « the maximum flow is equal to the minimum cut separating the related origin and target nodes »
- Δ(k) are the min-cuts separating the origin and target nodes of k
  - we consider the fully compressed expected flow from the DP traffic
  - cuts found with Edmonds-Karp's algorithm, Dinic's algorithm, and Boykov-Kolmogorov algorithm

$$\sum_{a \in \delta}(\gamma^{ka}_{f_{|F^d|}f_0} + \sum_{f \in \{f_0\} \cup F^d \setminus \{f_{|F^d|}\}} \gamma^{ka}_{ff+1}) \geq 1 \qquad \forall k \in K(s) : s \in S, \forall \delta \in \Delta(k)$$

Example : 100Gbs as expected compressed flow



Row Generation

Overall Idea

- Reduced MILP based on the original compact formulation
  - Only with:
    - split selection inequalities
    - dimensioning equations
    - packing inequalities
    - placement constraints
    - routing constraints
    - integrality constraints
- The reduced MILP can be up to 98% smaller

- Remained **isolation, capacity, and latency**
  - applied as **lazy constraints** within the branch-and-bound framework
    - if the current solution violates any lazy constraint, the latter is **added as cut to the reduced model**
  - parallelized framework
    - **each thread** is responsible for **searching** and **adding** the violated constraints

Numerical Results

Model Strengthening

- Test Setup
  - Cplex 12.10 : solver's pre-solving routines disabled
  - Time limit set to 3600 seconds
  - Three instance sizes (30 tests on each size)

Instance size	$\|V\|$	Graph density	$\|S\|$	Demands per slice	$\|F^d\|$	$\|F^c\|$
Tiny (T)	10	0.15	2	1	2	2
Small (S)	15	0.10	2	2	4	2
Medium (M)	20	variable	4	3	4	3

Linear Relaxation Gap

Instance Size	MILP	MILP + SB	MILP + LB	MILP + MC	MILP + SP	MILP + All
Tiny	28,50 %	28,50 %	12,12 %	28,50 %	28,50 %	12,12 %
Small	23,87 %	23,87 %	10,25 %	23,87 %	23,87 %	10,25 %

Final Gap

Instance Size	MILP	MILP + SB	MILP + LB	MILP + MC	MILP + SP	MILP + All
Tiny	8,87 %	5,50 %	3,50 %	5,75 %	6,12 %	3,12 %
Small	17,37 %	10,75 %	6,62 %	11,75 %	12,00 %	6,00 %

Numerical Results

Model Strengthening

- Lower-bound (LB) inequality
  - Better gap after solving the related LP
- Symmetry-breaking (SB), Min-Cut (MC), and Shortest Path (SP) inequalities
  - Improved final gap
- Best final gap with all proposed inequalities

Instance size	$\|V\|$	Graph density	$\|S\|$	Demands per slice	$\|F^d\|$	$\|F^c\|$
Tiny (T)	10	0.15	2	1	2	2
Small (S)	15	0.10	2	2	4	2
Medium (M)	20	variable	4	3	4	3

Linear Relaxation Gap

Instance Size	MILP	MILP + SB	MILP + LB	MILP + MC	MILP + SP	MILP + All
Tiny	28,50 %	28,50 %	12,12 %	28,50 %	28,50 %	12,12 %
Small	23,87 %	23,87 %	10,25 %	23,87 %	23,87 %	10,25 %

Final Gap

Instance Size	MILP	MILP + SB	MILP + LB	MILP + MC	MILP + SP	MILP + All
Tiny	8,87 %	5,50 %	3,50 %	5,75 %	6,12 %	3,12 %
Small	17,37 %	10,75 %	6,62 %	11,75 %	12,00 %	6,00 %

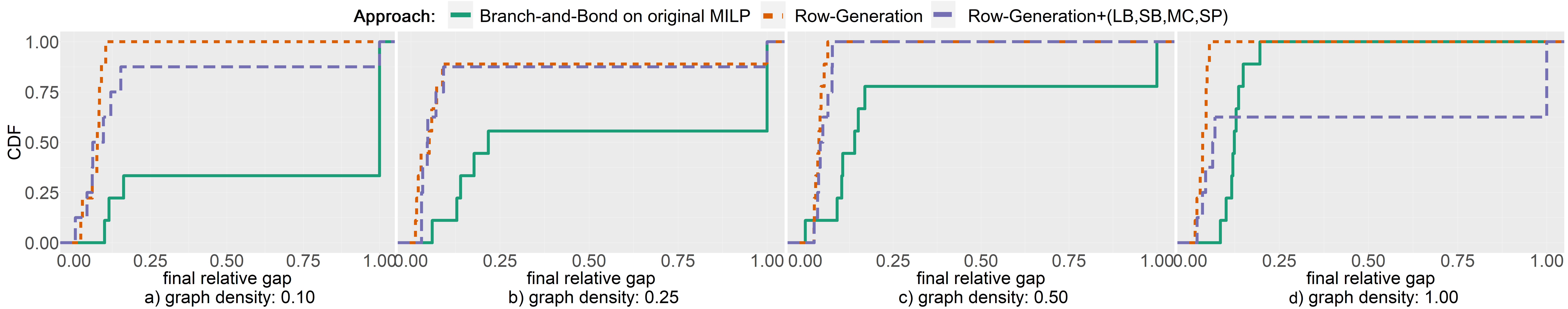
Numerical Results

Row Generation

- Medium-size instances: 30 tests with each graph (physical network) density
- Solver's pre-processing routines activated

- Row generation outperformed classic BB in all instance classes
  - final gap smaller than 10% in more than 90% of all instances

- Branch-and-Bound has better performance in high-density graphs
  - Outperformed Strengthened Reduced Model within the Row Generation framework



Heuristic Approaches for the NSDP

Math-Heuristic :: Relax-and-Fix

A Math-Heuristic for the NSDP

Algorithm 1: Math-heuristic for the NSDP

input : An NSDP instance $\mathbb{I}(G,S,F,N,C)$.
output: A solution to $\mathbb{I}$.

1 BestSolution, CurrentSolution $\leftarrow \emptyset$
2 **while** *a feasible solution to* $\mathbb{I}$ *is not found* **do**
3   chooseCUs()
4   **foreach** $s \in S$ **do**
5   **foreach** $k \in K(s)$ **do**
6   getPaths() ;
7   choosePaths() ;
8   selectSplit()
9   **while** *a feasible embedding is not found* ***or*** *maximal number of tries is reached* **do**
10   **if** $N \leftarrow$ *packNFSs()* ;
11   *is not feasible* **then** stop and go to step 3;
12   **else if** *embedNFs() fails* ***or*** *maximal number of tries is reached* **then** stop and go to step 3;
13   **if** *routing() fails;*
14   **then** stop and go to step 3 ;
15   **if** *CurrentSolution is feasible and cost(CurrentSolution) < cost(BestSolution)* **then**
16   BestSolution $\leftarrow$ CurrentSolution
17   **if** *rand()* $> \rho$ **then**
18   try to find another solution to $\mathbb{I}$ by going to step 3
19   **else**
20   **return** BestSolution

Overall Idea

- Decomposing the NSDP into several sub-problems
  - Split Selection
  - NFS-NF packing
  - NF-Node embedding
  - Traffic routing
- Input
  - a direct graph G (physical network) and capacities
  - a set S of slice requests with traffic demands K(s)
  - a set F of NFS types
  - a set N of potential host virtual functions
- Output
  - a virtual network for each slice request ensuring all technical constraints

A Math-Heuristic for the NSDP

Algorithm 1: Math-heuristic for the NSDP

input : An NSDP instance $\mathbb{I}(G,S,F,N,C)$.
output: A solution to $\mathbb{I}$.
1 BestSolution, CurrentSolution $\leftarrow \emptyset$
2 **while** *a feasible solution to* $\mathbb{I}$ *is not found* **do**
3 chooseCUs()
4 **foreach** $s \in S$ **do**
5 **foreach** $k \in K(s)$ **do**
6 getPaths() ; /* By Yen's algorithm */
7 choosePaths() ; /* See ILP (3)-(7) */
8 selectSplit()
9 **while** *a feasible embedding is not found* ***or*** *maximal number of tries is reached* **do**
10 **if** *$N \leftarrow$ packNFSs() ;*
11 *is not feasible* **then** stop and go to step 3;
12 **else if** *embedNFs() fails* ***or*** *maximal number of tries is reached* **then** stop and go to step 3;
13 **if** *routing() fails;*
14 **then** stop and go to step 3 ;
15 **if** *CurrentSolution is feasible and cost(CurrentSolution) $<$ cost(BestSolution)* **then**
16 BestSolution $\leftarrow$ CurrentSolution
17 **if** *rand() $> \rho$* **then**
18 try to find another solution to $\mathbb{I}$ by going to step 3
19 **else**
20 **return** BestSolution

Split Selection

- Create a set with the most centralized CUs
- Try to find a path for each traffic demand passing by the centralized CUs
- Chose the best path for each traffic demand
- Select the split setting for each slice

$$max \sum_{u,v \in V^h | u \neq v} \pi_{uv} z_{uv} \tag{3}$$

$$\sum_{p \in P(k)} x_p^k = 1 \qquad , \forall k \in K(s) : s \in S \tag{4}$$

$$\sum_{k \in K(s)} \sum_{p \in P(k)} \lambda_{uv}^p x_p^k = z_{uv} \qquad , \forall u,v \in V^h | u \neq v \tag{5}$$

$$x_p^k \in \{0,1\} \qquad , \forall k \in K(s) | s \in S, \forall p \in P(k) \tag{6}$$

$$z_{uv} \in \mathbb{N}_0 \qquad , \forall u,v \in V^h | u \neq v \tag{7}$$

A Math-Heuristic for the NSDP

Algorithm 1: Math-heuristic for the NSDP

```
   input : An NSDP instance 𝕀(G,S,F,N,C).
   output: A solution to 𝕀.
1  BestSolution, CurrentSolution ← ∅
2  while a feasible solution to 𝕀 is not found do
3      chooseCUs()
4      foreach s ∈ S do
5          foreach k ∈ K(s) do
6              getPaths() ;            /* By Yen's algorithm */
7          choosePaths() ;             /* See ILP (3)-(7) */
8      selectSplit()
9      while a feasible embedding is not found or
        maximal number of tries is reached do
10         if N ← packNFSs() ;
11         is not feasible then stop and go to step 3;
12         else if embedNFs() fails or maximal number of
            tries is reached then stop and go to step 3;
13     if routing() fails;
14      then stop and go to step 3 ;
15     if CurrentSolution is feasible and
        cost(CurrentSolution) < cost(BestSolution) then
16         BestSolution ← CurrentSolution
17         if rand() > ρ then
18             try to find another solution to 𝕀 by going to
                step 3
19         else
20             return BestSolution
```

NFS-NF packing

- Create conflict graph *C*
  - each built NFS is now a vertex
  - isolation and capacity constraints as edges
- Color the the conflict graph *C*
  - several times in parallel
    - Greed approach with random vertex ordering
  - compare each try to the max-clique number (theorical LB)
  - take the best coloring
- All vertex (NFSs) with the same color are packed into the same NF

A Math-Heuristic for the NSDP

Algorithm 1: Math-heuristic for the NSDP

input : An NSDP instance $\mathbb{I}(G,S,F,N,C)$.
output: A solution to $\mathbb{I}$.
1 BestSolution, CurrentSolution $\leftarrow \emptyset$
2 **while** *a feasible solution to* $\mathbb{I}$ *is not found* **do**
3   chooseCUs()
4   **foreach** $s \in S$ **do**
5   **foreach** $k \in K(s)$ **do**
6   getPaths() ;   `/* By Yen's algorithm */`
7   choosePaths() ;   `/* See ILP (3)-(7) */`
8   selectSplit()
9   **while** *a feasible embedding is not found* ***or*** *maximal number of tries is reached* **do**
10   **if** $N \leftarrow$ *packNFSs()* ;
11   *is not feasible* **then** stop and go to step 3;
12   **else if** *embedNFs() fails* ***or*** *maximal number of tries is reached* **then** stop and go to step 3;
13   **if** *routing() fails;*
14   **then** stop and go to step 3 ;
15   **if** *CurrentSolution is feasible and cost(CurrentSolution)* $<$ *cost(BestSolution)* **then**
16   BestSolution $\leftarrow$ CurrentSolution
17   **if** *rand()* $> \rho$ **then**
18   try to find another solution to $\mathbb{I}$ by going to step 3
19   **else**
20   **return** BestSolution

NF-physical node embedding

- Solved as a Bin-Packing Problem
  - Each NF is now an object
  - Each physical node is now a bin
- Randomly solved several times in parallel
  - Return the best packing

A Math-Heuristic for the NSDP

Algorithm 1: Math-heuristic for the NSDP

```
   input : An NSDP instance 𝕀(G,S,F,N,C).
   output: A solution to 𝕀.
1  BestSolution, CurrentSolution ← ∅
2  while a feasible solution to 𝕀 is not found do
3      chooseCUs()
4      foreach s ∈ S do
5          foreach k ∈ K(s) do
6              getPaths() ;        /* By Yen's algorithm */
7          choosePaths() ;         /* See ILP (3)-(7) */
8      selectSplit()
9      while a feasible embedding is not found or
        maximal number of tries is reached do
10         if N ← packNFSs() ;
11         is not feasible then stop and go to step 3;
12         else if embedNFs() fails or maximal number of
            tries is reached then stop and go to step 3;
13     if routing() fails;
14      then stop and go to step 3 ;
15     if CurrentSolution is feasible and
        cost(CurrentSolution) < cost(BestSolution) then
16         BestSolution ← CurrentSolution
17         if rand() > ρ then
18             try to find another solution to 𝕀 by going to
                step 3
19         else
20             return BestSolution
```

Traffic Routing

- Create a set of feasible paths for each flow
  - By Yen's algorithm
  - Randomly chose a path for each flow
  - Verify overall resource consumption
- Run several times if necessary or until stop criteria is reached

A Math-Heuristic for the NSDP

Algorithm 1: Math-heuristic for the NSDP

```
input  : An NSDP instance 𝕀(G,S,F,N,C).
output: A solution to 𝕀.
1  BestSolution, CurrentSolution ← ∅
2  while a feasible solution to 𝕀 is not found do
3      chooseCUs()
4      foreach s ∈ S do
5          foreach k ∈ K(s) do
6              getPaths() ;          /* By Yen's algorithm */
7          choosePaths() ;           /* See ILP (3)-(7) */
8      selectSplit()
9      while a feasible embedding is not found or
        maximal number of tries is reached do
10         if N ← packNFSs() ;
11         is not feasible then stop and go to step 3;
12         else if embedNFs() fails or maximal number of
            tries is reached then stop and go to step 3;
13     if routing() fails;
14      then stop and go to step 3 ;
15     if CurrentSolution is feasible and
        cost(CurrentSolution) < cost(BestSolution) then
16         BestSolution ← CurrentSolution
17         if rand() > ρ then
18             try to find another solution to 𝕀 by going to
                step 3
19         else
20             return BestSolution
```

Returning the best solution

- Tabu Search-inspired
  - Save the solution if it is feasible
  - Try to find another solution if:
    - Current solution is not feasible
    - Stop criteria is not reached (overall runtime related)

A Relax-and-Fix algorithm for the NSDP with dedicated NFs

Overall Idea

- Starting without NFS-NF packing sub-problem
- Repetitively solve the proposed (M)ILP
  - Only a few integer/binary variables
  - Relaxing and Fixing most of the remaining integer/binary variables
- Solve NFS-NF packing sub-problem with Vertex Coloring
  - Conflict graph

Algorithm 1 Relax-and-Fix

input : An NSDP-DNF instance $\mathbb{I}(G, S, F, N, C)$ and a pacing strategy ρ.

output: A solution (if there exists one) to $\mathbb{I}$.

$N^* \leftarrow \{n_1\}$
Create model $\mathbb{M}$;
Relax all integrality constraints in $\mathbb{M}$
Order S ;
p $\leftarrow \rho$;
Let S^* be the set of embedded NSs: set it to a set
while $S^* \neq S$ ***and feasibility condition is respected*** **do**
    Set IntSlices to the first ρ slices in $S \backslash S^*$
    foreach *s in IntSlices* **do**
        enforce integrality on all related variables in $\mathbb{M}$
    solve $\mathbb{M}$
    if *a feasible solution is found* **then**
        foreach *s in IntSlices* **do**
            Fix values on all related variables in $\mathbb{M}$
        Add IntSlices to S^*
        p $\leftarrow \rho$
    else if $S^* \neq \emptyset$ **then**
        Remove the last embedded slice from S^*
        p$\leftarrow$p+1;
    else
        Stop: no feasible solution exists
if $\mathbb{M}$ *is feasible* **then**
    PackNFSs() ;
    return the complete solution to $\mathbb{I}$
else
    return no solution

A Relax-and-Fix algorithm for the NSDP with dedicated NFs

Pacing Strategy

- Slice-based decomposition
- Slice ordering: capacity, latency, randomly

On each iteration

- Restore integrality constraints on all variables related to a sub-set S* of slices
- Solve (M)ILP
- Fix found values on all variables related to S*
- Create new S*
- Repeat

Algorithm 1 Relax-and-Fix

```
input  : An NSDP-DNF instance 𝕀(G, S, F, N, C) and a pacing strategy ρ.
output : A solution (if there exists one) to 𝕀.
N* ← {n_1}
Create model 𝕄;
Relax all integrality constraints in 𝕄
Order S ;
p ← ρ ;
Let S* be the set of embedded NSs: set it to a set
while S* ≠ S and feasibility condition is respected do
    Set IntSlices to the first ρ slices in S\S*
    foreach s in IntSlices do
        enforce integrality on all related variables in 𝕄
    solve 𝕄
    if a feasible solution is found then
        foreach s in IntSlices do
            Fix values on all related variables in 𝕄
        Add IntSlices to S*
        p ← ρ
    else if S* ≠ ∅ then
        Remove the last embedded slice from S*
        p←p+1;
    else
        Stop: no feasible solution exists
if 𝕄 is feasible then
    PackNFSs() ;
    return the complete solution to 𝕀
else
    return no solution
```

Numerical Results

Different instance classes

- Strict versus Relaxed latency constraints
- Tight versus Moderate capacity on physical network
- Strong versus Weak isolation constraints

Different instance sizes

Instance size	$\lvert V \rvert$	Graph Density*	$\lvert S \rvert$	$\lvert K \rvert$	$\lvert F^d \rvert$	$\lvert F^c \rvert$
Tiny (T)	10	0.15	2	1	2	2
Small (S)	15	0.10	2	2	4	2
Medium-Small (SM)	20	0.15	4	3	4	3
Medium (M)	25	0.15	4	8	6	4
Medium-Big (MB)	30	0.20	4	8	6	6
Big (B)	35	0.20	8	8	8	6
Extra-Big (EB)	40	0.25	8	8	8	8

* Ratio between exiting and theoretically possible number of arcs.

Numerical Results

Math-Heuristic

- **Tiny** instances with **moderate capacity** constraints were solved **faster**
  - less than 1 second in general

- More than **80%** of tiny instances had a gap smaller then **2%**

- We did not observe any significant impact from different instance classes on bigger instances

Numerical Results

Math-Heuristic

- Feasible solution found for all instance sizes
- Faster than MILP
- Better gap on bigger instances
  - Medium-big instances : from 36% to 4%

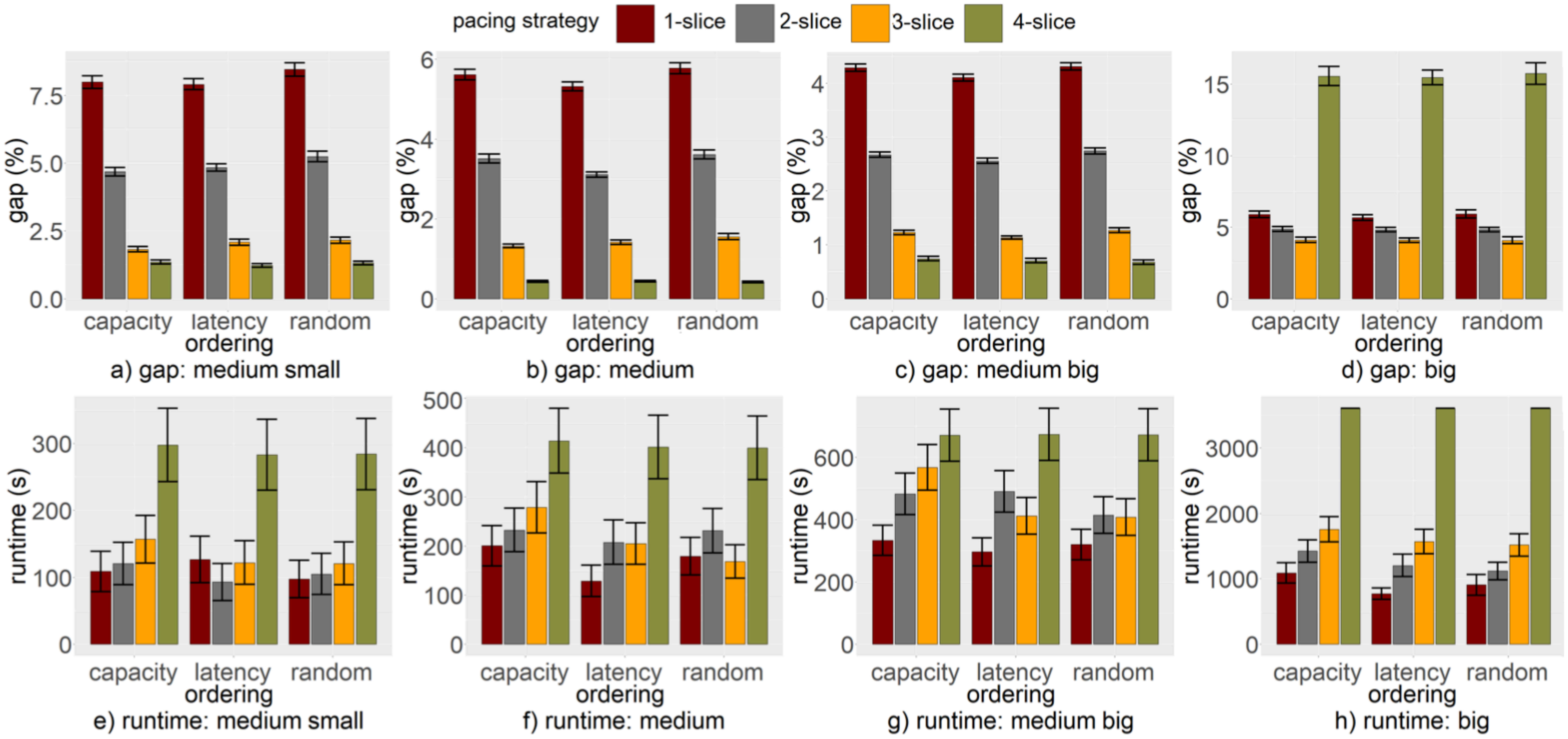
Instance Size	MILP		Math-Heuristic	
	Runtime (s)	Gap (%)	Runtime (s)	Gap (%)
Medium-Small	2165 ± 364	0	**732 ± 87**	3.2 ± 0.5
Medium	3600*	5.7 ± 2.1	**803± 122**	4.5±0.7
Medium-Big	3600*	36.3 ± 4.8	**894 ± 109**	**4.3±1.2**
Big	3600*	**	997 ± 84	**8.2±3.6**
Extra-Big	3600*	**	**1245 ± 241**	11.5±2.4

* Time limit reached. ** No feasible solution was found.

Numerical Results

Relax-and-Fix

- Better solutions quality with bigger pacing strategies
- Runtime increases as the pacing strategy gets bigger
    - bigger increases from 3-slice pacing to 4-slice pacing in smaller instances



Numerical Results

Relax-and-Fix

- Random ordering and 1-slice pacing strategies
- Runtime : outperformed ILP on all instance sizes
- Good solution even for big instances
- No significant impact from different instance classes

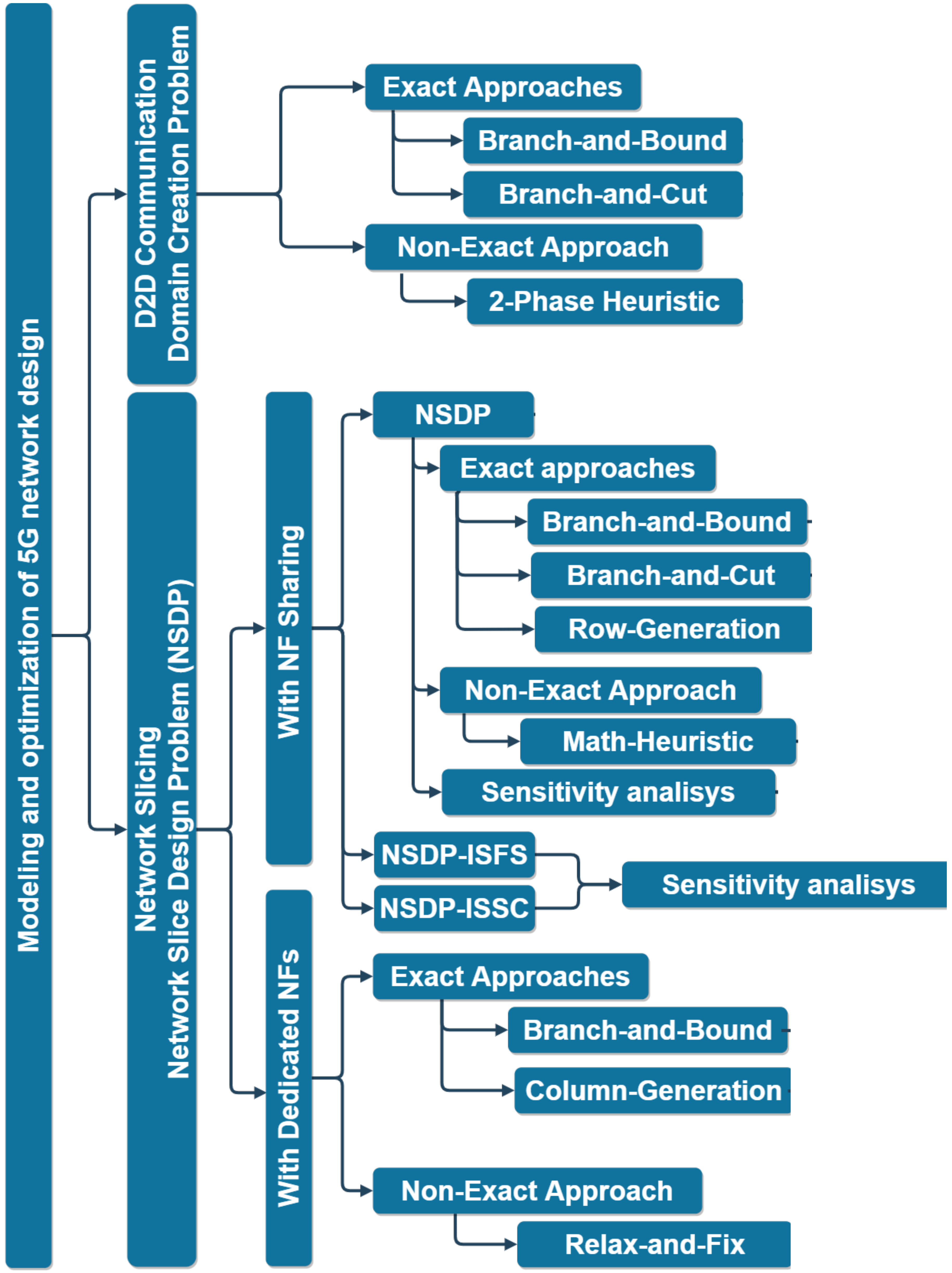
	ILP		Relax-and-Fix	
Instance Size	**Gap (%)**	**Runtime (s)**	**Gap (%)**	**Runtime (s)**
Tiny	0.0	5 ± 3	$\mathbf{1.14 \pm 1.1}$	$\mathbf{0.03\pm0.01}$
Small	0.0	248 ± 49	$\mathbf{1.65 \pm 0.3}$	$\mathbf{0.23\pm0.1}$
Medium Small	26.5 ± 2.3	3600*	8.44 ± 0.2	$\mathbf{97\pm28}$
Medium	48.4 ± 28.8	3600*	$\mathbf{5.77\pm0.1}$	$\mathbf{178\pm37}$
Medium Big	82.8 ± 32.2	3600*	$\mathbf{4.31\pm0.06}$	$\mathbf{319\pm49}$
Big	**	3600*	$\mathbf{5.90\pm0.1}$	907 ± 158

* Time limit reached. ** No feasible solution was found.

Concluding Remarks

Summary :: Perspectives

Concluding Remarks



Summing Up

- Modeling and optimization of 5G network design
- Domain Creation problem with D2D communication
  - Exact and Non-Exact Approaches
- k Slice Design
  - ...ent Variants
  - Several approaches
- Contributions
  - 4 published papers
  - 2 submitted papers
  - 1 paper in preparation

Perspectives and Future Works

- **New NSDP Variants**
  - with Multi-Access Edge Computing for RAN slice subnets
  - service-aware objective functions
  - availability constraints

- **Relax-and-Fix**
  - pacing strategies based on NFs, traffic demands, variables, etc…

- **Clustering pre-processing**
  - based on geographical zone, required service, etc…
  - Solve smaller clustered NSDP instances in parallel

THANK YOU