A Hybrid Virtual Network Function Placement Strategy for Maximizing the Profit of Network Service Deployment over Dynamic Workload

Authors: Chi-Chen Yang, Jerry Chou
Computer Science Dept.
National Tsing Hua University
Hsinchu, Taiwan
6/21/2021 @ SNTA'21

Outline

- Motivation & Objective
  - Introduction of Network Function Virtualization
  - Placement Problems and Challenges
  - Existing Approaches
- Methodology
- Experimental Evaluations
- Conclusion

Outline

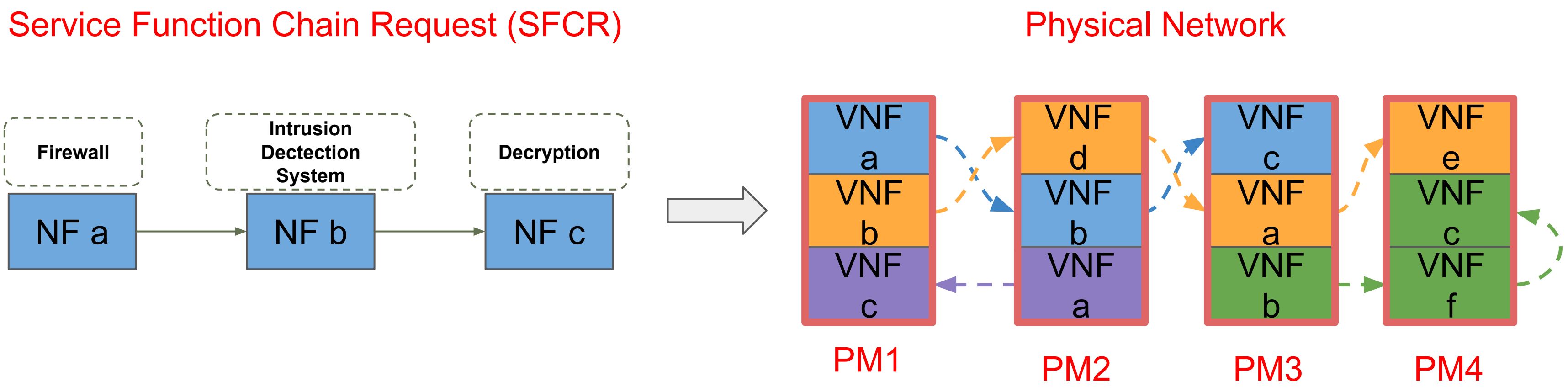
- Motivation & Objective
  - Introduction of Network Function Virtualization
  - Placement Problems and Challenges
  - Existing Approaches
- Methodology
- Experimental Evaluations
- Conclusion

Network Function Virtualization (NFV)

- Network function (NF)
  - Firewall, load balancer, etc
- Virtualization
  - Past: specific hardwares
  - Now: software processes
    - Can be containerized and hosted on commodity servers
- For network operators
  - Increase the flexibility of deployment and maintenance
  - Reduce the cost of infrastructure construction

Placement Problems

- From a network operator's point of view
- Network service requests from users
  - Service function chain (SFC): a list of network functions within a specified order



Placement Problems and Challenges

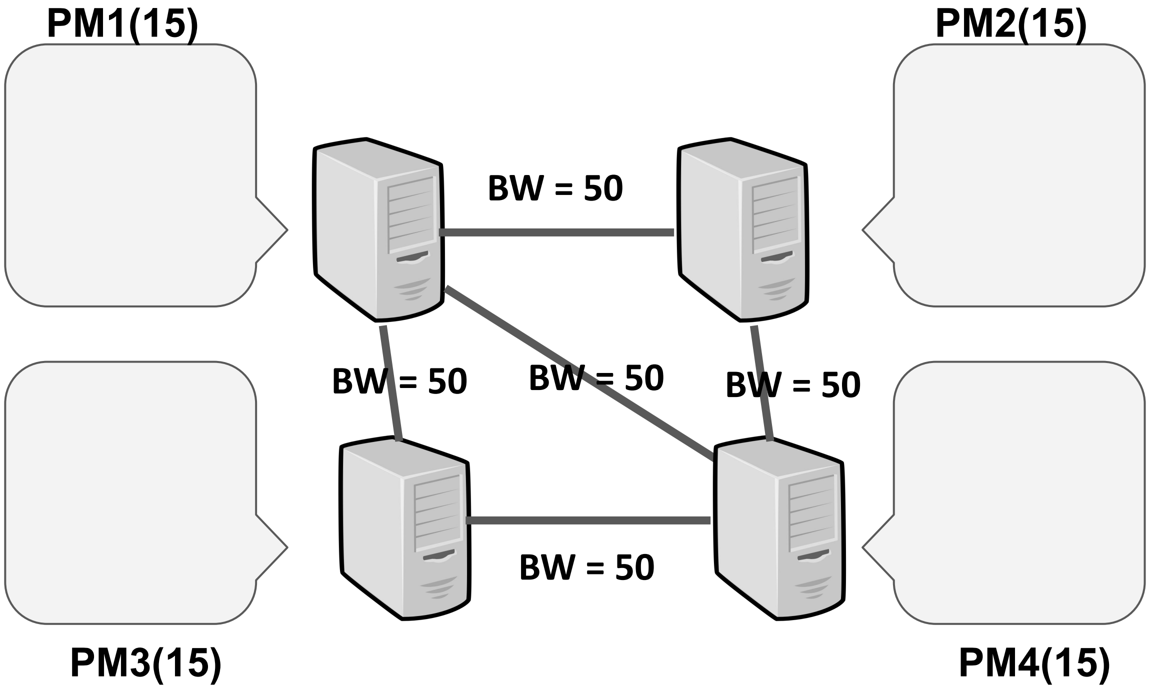
- Do the decisions
  - VNF placement
  - NF mapping
  - SFC routing
  - Accept/Reject SFCR
- Objectives
  - Maximize service provider's revenue: traffic demand of SFC, end-to-end delay of SFC
  - Minimize service operation cost: resource and energy consumption
  - Minimize reconfiguration cost: service interruption, traffic routing, VNF migration
- Constraints
  - Limited computing capacity on physical nodes
  - Limited bandwidth on network links
  - Service quality requirement

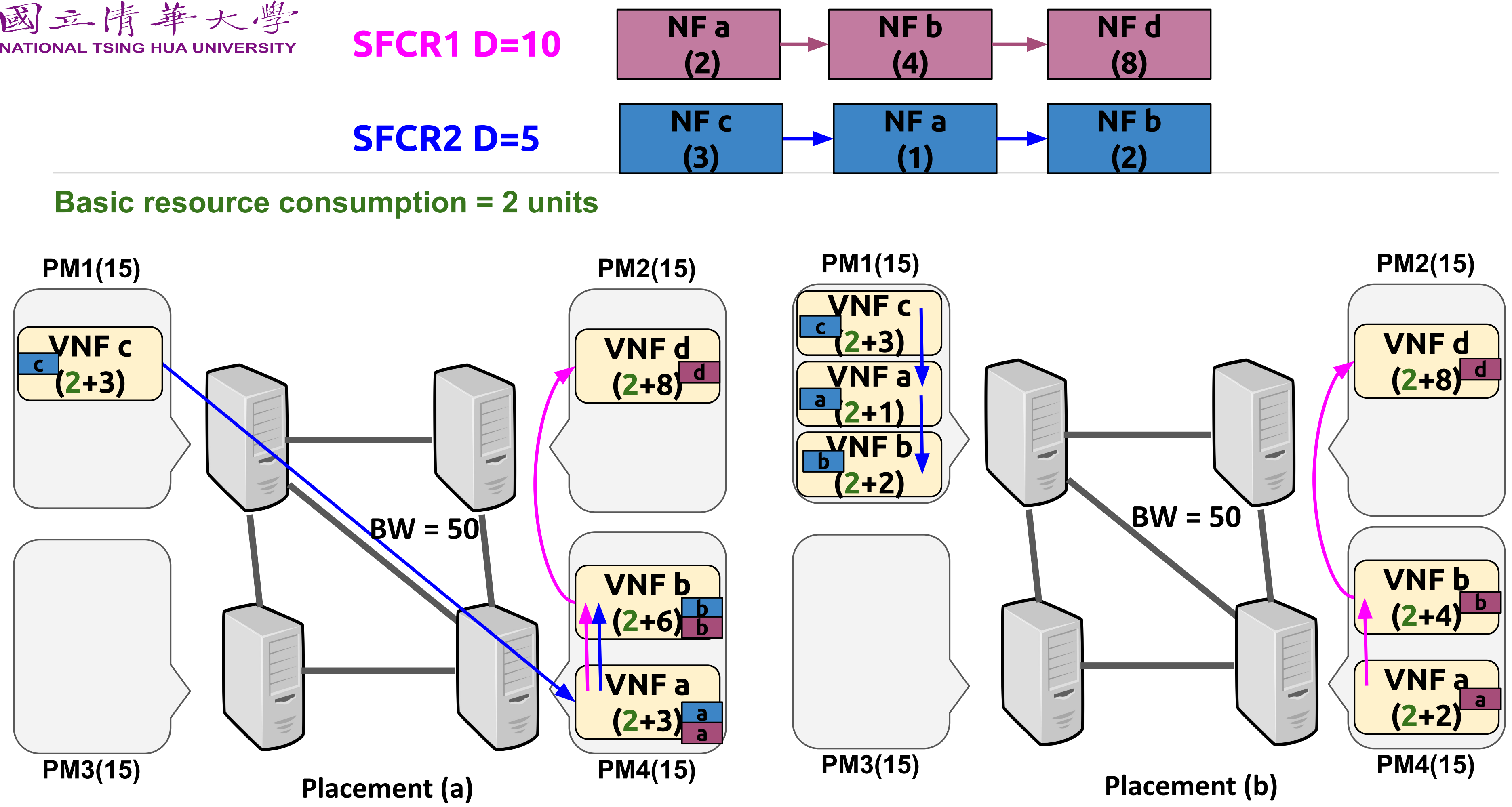
SFCR1 D=10

NF a (2) → NF b (4) → NF d (8)

SFCR2 D=5

NF c (3) → NF a (1) → NF b (2)

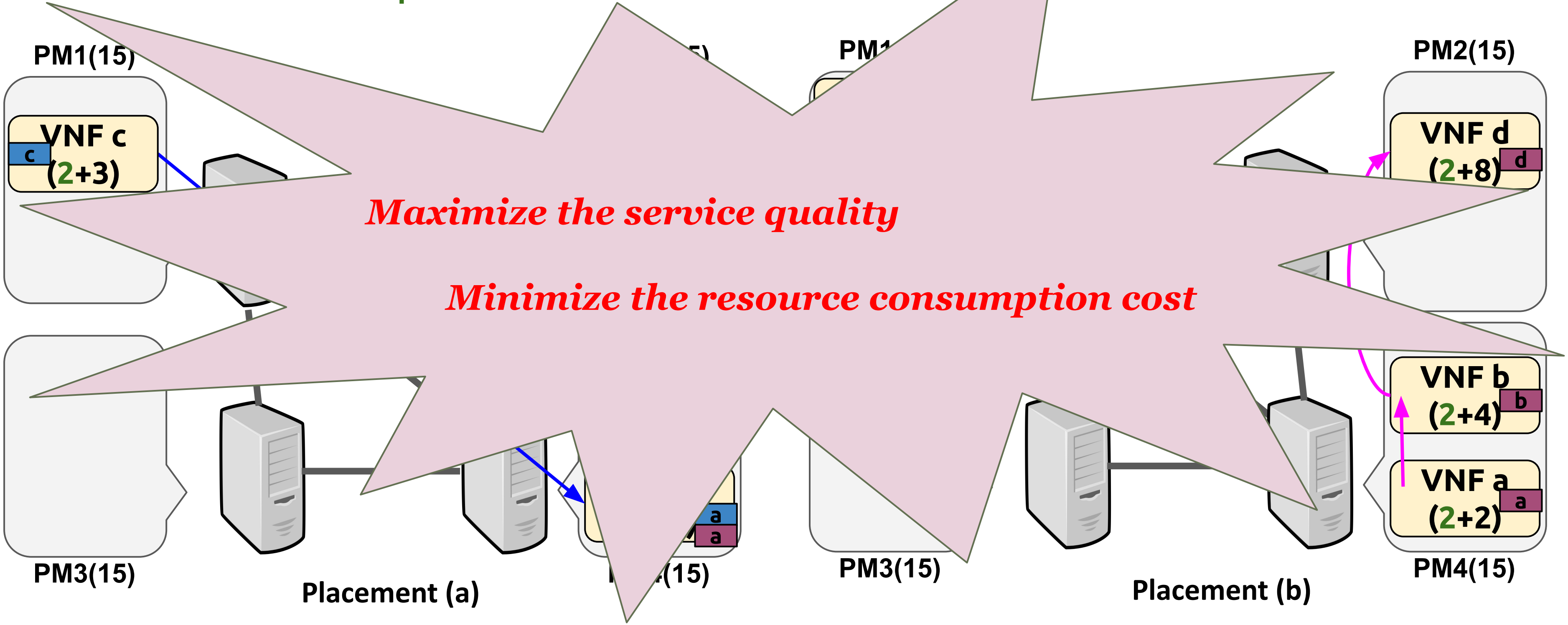



SFCR1 D=10
NF a (2)
NF b (4)
NF d (8)
SFCR2 D=5
NF c (3)
NF a (1)
NF b (2)
Basic resource consumption = 2 units
PM1(15)
VNF c
c
(2+3)
PM2(15)
VNF d
(2+8)
d
BW = 50
VNF b
(2+6)
b
b
VNF a
(2+3)
a
a
PM3(15)
PM4(15)
Placement (a)
PM1(15)
VNF c
c
(2+3)
VNF a
a
(2+1)
VNF b
b
(2+2)
PM2(15)
VNF d
(2+8)
d
BW = 50
VNF b
(2+4)
b
VNF a
(2+2)
a
PM3(15)
PM4(15)
Placement (b)

SFCR1 D=10: NF a (2) → NF b (4) → NF d (8)

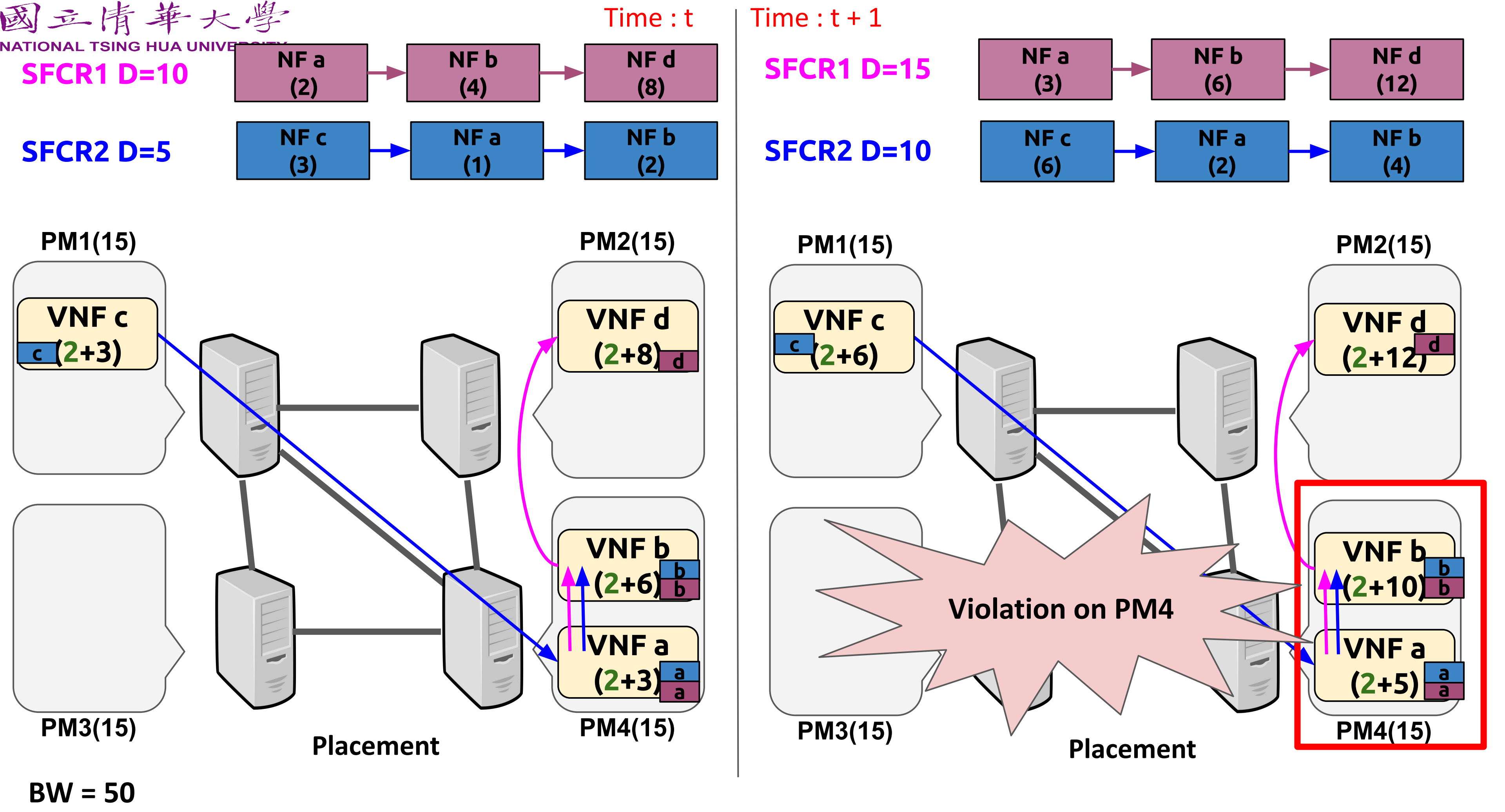
SFCR2 D=5: NF c (3) → NF a (1) → NF b (2)

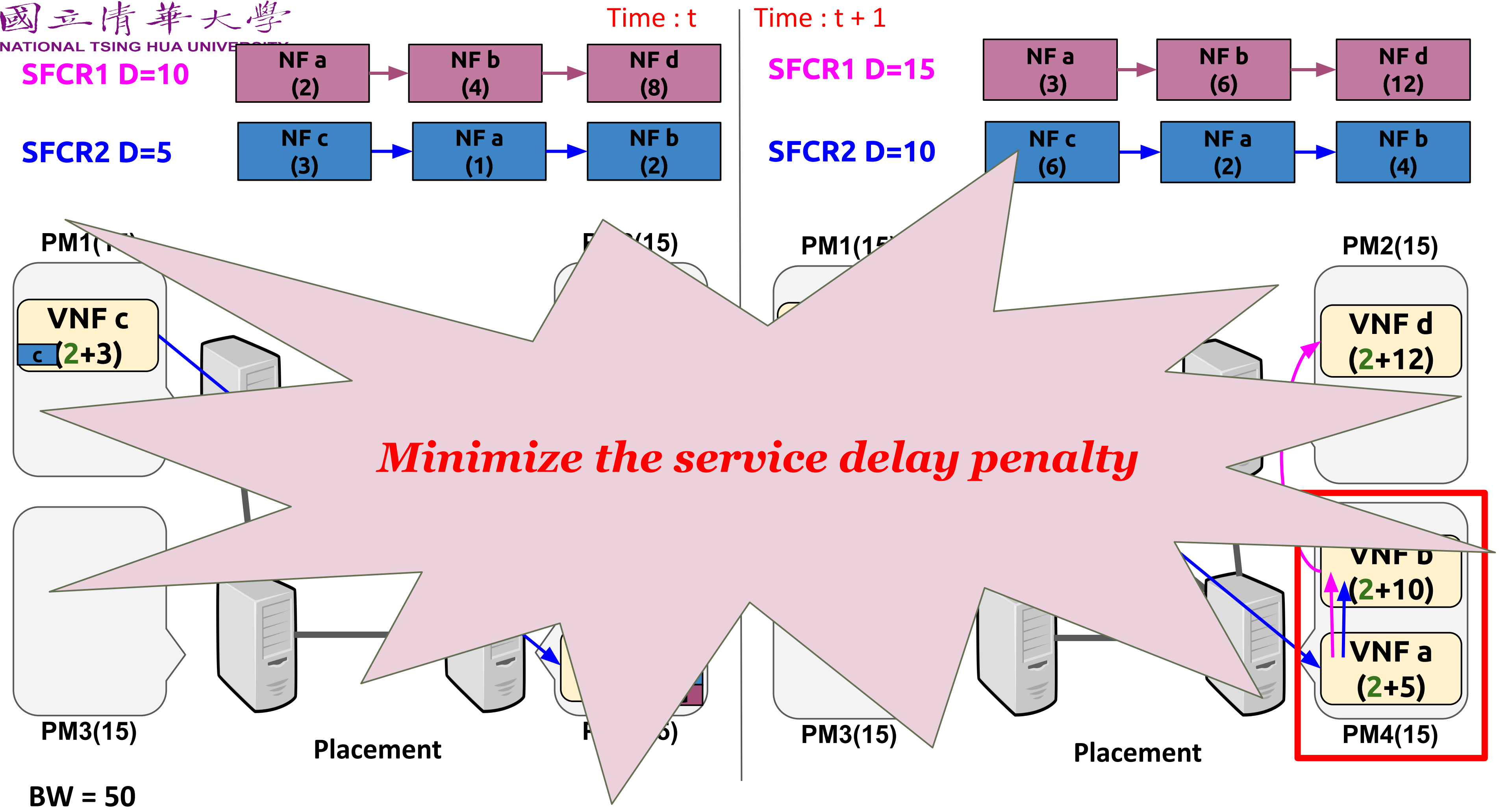
Basic resource consumption = 2 units



Placement Problems and Challenges

- Dynamic traffic demand of network service
- Computing resource
  - Vertical scaling
    - Leverage idle resources in the physical machine
  - Horizontal scaling
    - Initialize a virtual machine
      - Boot the operating system
      - Initialize the correspoding network function application
    - Migrate the state information of VNF
- Bandwidth resource
  - Reroute traffic


Time : t
SFCR1 D=10
NF a (2)
NF b (4)
NF d (8)
SFCR2 D=5
NF c (3)
NF a (1)
NF b (2)
PM1(15)
VNF c
c (2+3)
PM2(15)
VNF d
(2+8) d
VNF b
(2+6) b b
VNF a
(2+3) a a
PM3(15)
PM4(15)
Placement
BW = 50
Time : t + 1
SFCR1 D=15
NF a (3)
NF b (6)
NF d (12)
SFCR2 D=10
NF c (6)
NF a (2)
NF b (4)
PM1(15)
VNF c
c (2+6)
PM2(15)
VNF d
(2+12) d
VNF b
(2+10) b b
VNF a
(2+5) a a
Violation on PM4
PM3(15)
PM4(15)
Placement


Time : t
SFCR1 D=10
NF a (2)
NF b (4)
NF d (8)
SFCR2 D=5
NF c (3)
NF a (1)
NF b (2)
VNF c
c (2+3)
PM3(15)
Placement
Time : t + 1
SFCR1 D=15
NF a (3)
NF b (6)
NF d (12)
SFCR2 D=10
NF c (6)
NF a (2)
NF b (4)
PM2(15)
VNF d (2+12)
VNF b (2+10)
VNF a (2+5)
PM3(15)
PM4(15)
Placement
Minimize the service delay penalty

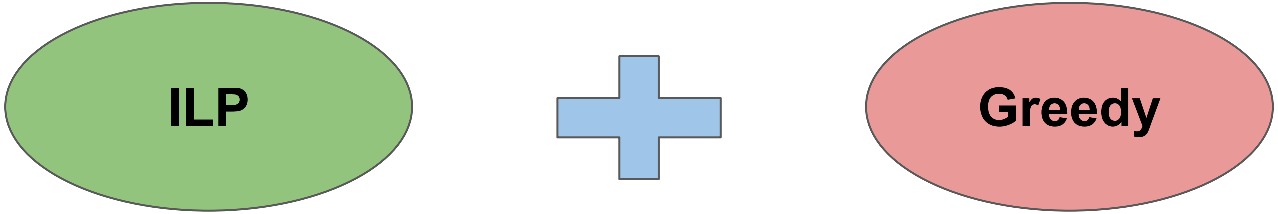
BW = 50

Existing Approaches

- **ILP**: formulate the problem as an Integer Linear Programming (ILP) Problem
  - Pros:
    - Find the optimal placement
  - Cons:
    - Time-consuming
      - Infeasible for large scale
      - Infeasible for dynamic workloads
- **Greedy**: propose huristic algorithms
  - Pros:
    - Fast
  - Cons:
    - Find an approximate solution

Proposed approach: hybrid method

- Combine two approaches



Pros:
- Find the optimal placement

Pros:
- Fast

Challenges of Hybrid

- When and how to use the ILP ?
- When and how to use the Greedy algorithm ?
- How to solve the time-consuming problem of ILP ?
- How to solve the problem of dynamic traffic ?

Outline

- Motivation & Objective
  - Introduction of Network Function Virtualization
  - Placement Problems and Challenges
  - Existing Approaches
- Methodology
- Experimental Evaluations
- Conclusion

Hybrid Design

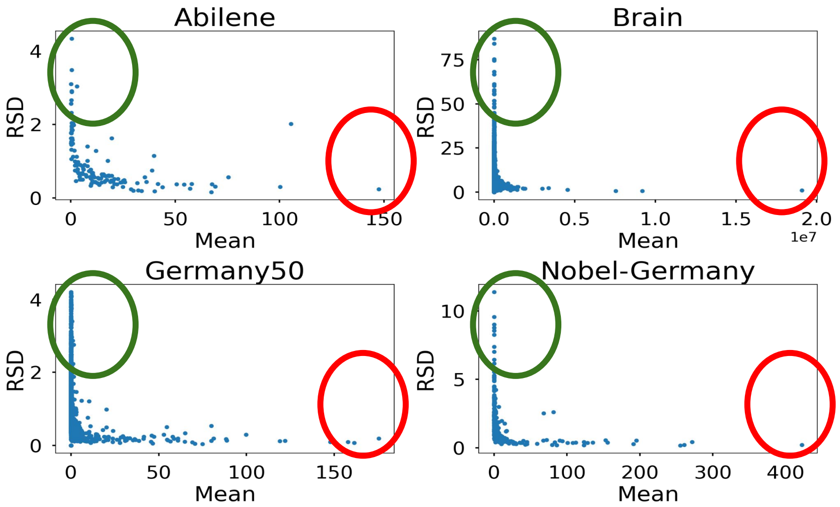
- To solve the time-consuming problem of ILP
  - Reduce the problem size of ILP
    - Classify SFCRs into stable and unstable
- The placement of stable SFCRs
  - ILP processes
    - The number of stable SFCRs is less than the number of unstable SFCRs
    - Obtain the maximum profit from heavy workload traffic
- The placement of unstable SFCRs
  - Greedy processes
    - More likely to be migrated over time due to traffic variations
- Place stable SFCRs first, not unstable SFCRs
  - Avoid resource fragmentation

Traffic Analysis

- Traffic :
  - With a heavier workload (mean) -> a lower relative standard deviation (RSD)
  - With a higher relative standard deviation (RSD) -> a lighter workload (mean)
- Most of the traffic has a small workload, and a few traffic has a larger workload.

Stable

Unstable

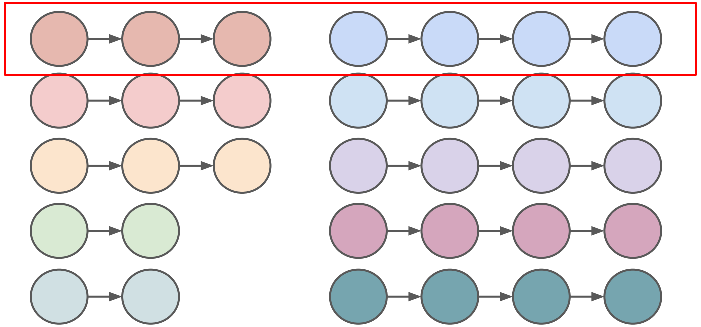


Workflow of Hybrid

First time interval: Initial Placement

ILP:
stable SFCRs

10 SFCRs (2 stable, 8 unstable)



stable SFCRs

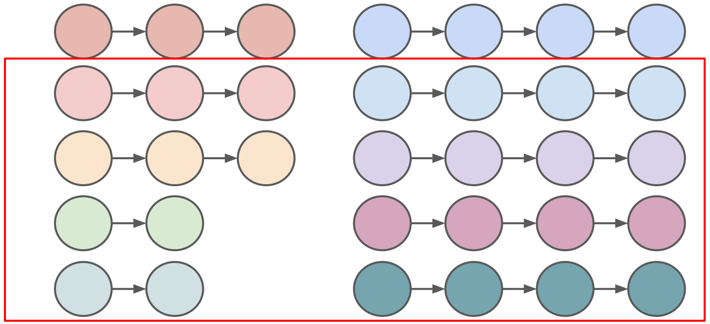
Workflow of Hybrid

First time interval: Initial Placement

ILP:
stable SFCRs

Greedy placement:
unstable SFCRs

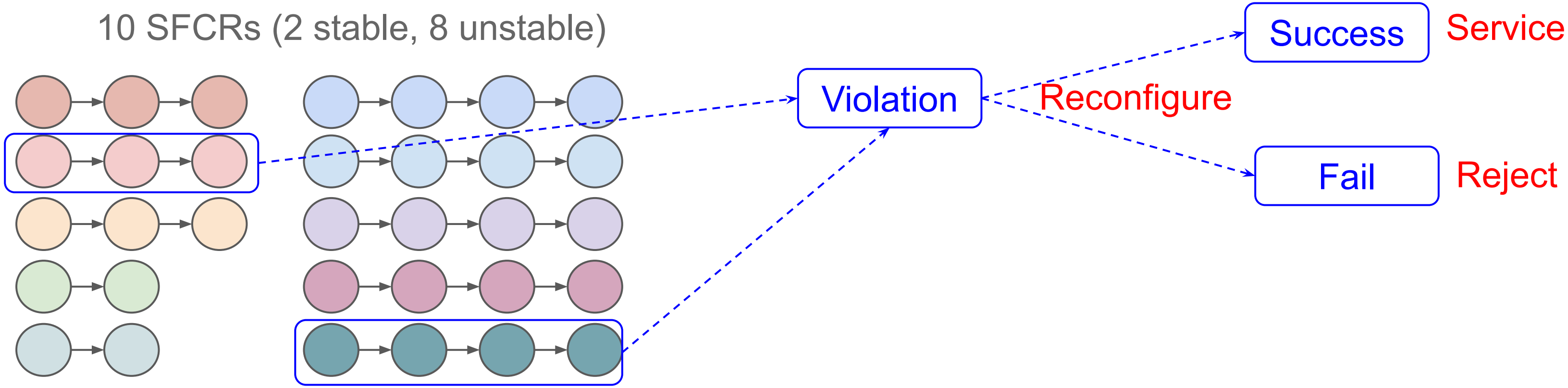
10 SFCRs (2 stable, 8 unstable)



unstable SFCRs

Workflow of Hybrid

10 SFCRs (2 stable, 8 unstable)



ILP Problem Formulation

- Objective function

maximize $netprofit = Revenue - \alpha * CC - \beta * DC - \gamma * RC$

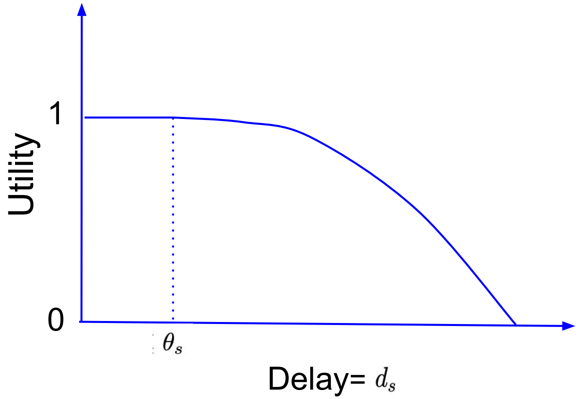
- Revenue

$$d_s = \sum_{i=0}^{|F|-1} \sum_{n=0}^{|N|-1} \sum_{m=0}^{|N|-1} y_{s,i,n,m}^{t}, \forall s \in S$$

$$Utility(d_s) = \begin{cases} 1, & \text{if } d_s \le \theta_s \\ ((-1 * exp(\epsilon * (d_s) + \eta)/\zeta), & \text{otherwise} \end{cases}$$

$$Revenue = \sum_{s=0}^{|S|-1} D_s * Utility(d_s)$$

SFCR traffic



ILP Problem Formulation

- Objective function

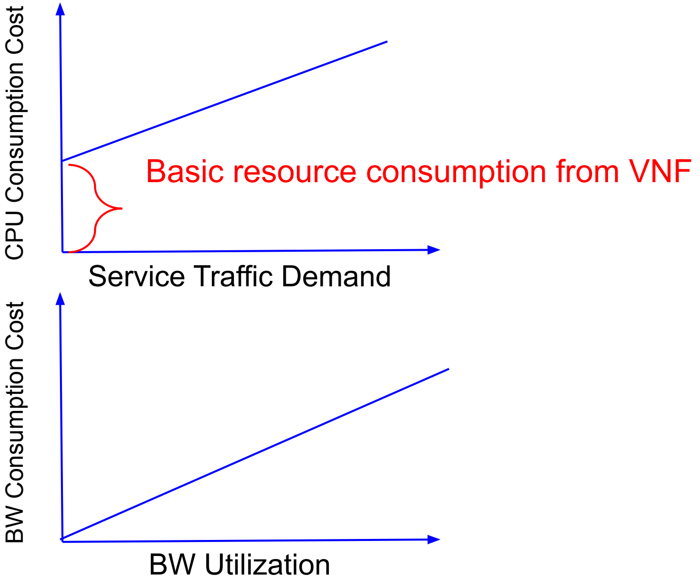
maximize $$netprofit = Revenue - \alpha * CC - \beta * DC - \gamma * RC$$

- Recource Consumption Cost (CC)

$$CPU = \sum_{s=0}^{|S|-1} \sum_{i=0}^{|F|-1} \sum_{n=0}^{|N|-1} FC_{s,i} * x^{t}_{s,i,n} + \sum_{n=0}^{|N|-1} \sum_{f=0}^{|F|-1} \phi_f * z^{t}_{f,n}$$

$$BW = \sum_{s=0}^{|S|-1} \sum_{i=0}^{|F|-2} \sum_{n=0}^{|N|-1} \sum_{m=0}^{|N|-1} D_s * y^{t}_{s,i,n,m}, n \neq m$$

$$CC = \omega * CPU + (1 - \omega) * BW$$

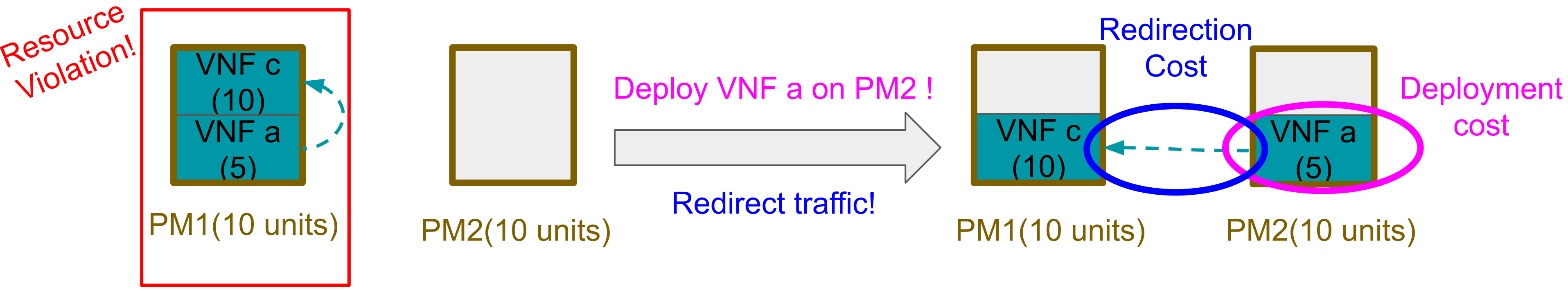


ILP Problem Formulation

- Objective function : maximize $netprofit = Revenue - \alpha * CC - \beta * DC - \gamma * RC$
- Deployment Cost (DC) & Redirection Cost (RC)

$$DC = \sum_{n=0}^{|N|-1} \sum_{f=0}^{|F|-1} \Gamma_f * (z_{f,n}^{t} - z_{f,n}^{t-1}) \qquad z_{f,n}^{t} = \begin{cases} 1, & \text{if } \sum_{s=0}^{|S|-1} \sum_{i=0}^{|F|-1} x_{s,i,n}^{t} * V_{s,i,f} \geq 1 \\ 0, & \text{otherwise} \end{cases}, f \in F, n \in N, t \in T$$

$$RC = \sum_{s=0}^{|S|-1} D_s * r_s^t \qquad r_s^t = \begin{cases} 1, & \text{if } \sum_{i=0}^{|F|-1} \sum_{n=0}^{|N|-1} |x_{s,i,n}^{t} - x_{s,i,n}^{t-1}| > 0 \\ & \text{or } \sum_{i=0}^{|F|-2} \sum_{n=0}^{|N|-1} \sum_{m=0}^{|N|-1} |y_{s,i,n,m}^{t} - y_{s,i,n,m}^{t-1}| > 0 \\ 0, & \text{otherwise} \end{cases}$$



ILP Problem Constraints

- Resource contraints

$$\sum_{s=0}^{|S|-1}\sum_{i=0}^{|F|-1} FC_{s,i} * x^{t}_{s,i,n} + \sum_{f=0}^{|F|-1} \phi_f * z^{t}_{f,n} <= C_n * (1-\Phi), n \in N, t \in T$$

$$\sum_{s=0}^{|S|-1}\sum_{i=0}^{|F|-2} FB_{s,i} * y^{t}_{s,i,n,m} <= BW_{n,m} * (1-\Psi)$$

- NF mapping constraint

$$\sum_{n=0}^{|N|-1} x^{t}_{s,i,n} <= 1, s \in S, t \in T$$

- Traffic contraints

$$\sum_{n=0}^{|N|-1} y^{t}_{s,i,m,n} <= 1, s \in S, n \in N, m \in N, n \neq m, t \in T$$

$$\sum_{n=0}^{|N|-1} y^{t}_{s,i,n,m} <= 1, s \in S, n \in N, m \in N, n \neq m, t \in T$$

$$\sum_{n=0}^{|N|-1} y^{t}_{s,i,n,m} - \sum_{n=0}^{|N|-1} y^{t}_{s,i,m,n} = x^{t}_{s,i,n} - x^{t}_{s,i+1,n}$$

$$, s \in S, n \in N, m \in S, n \neq m, t \in T$$

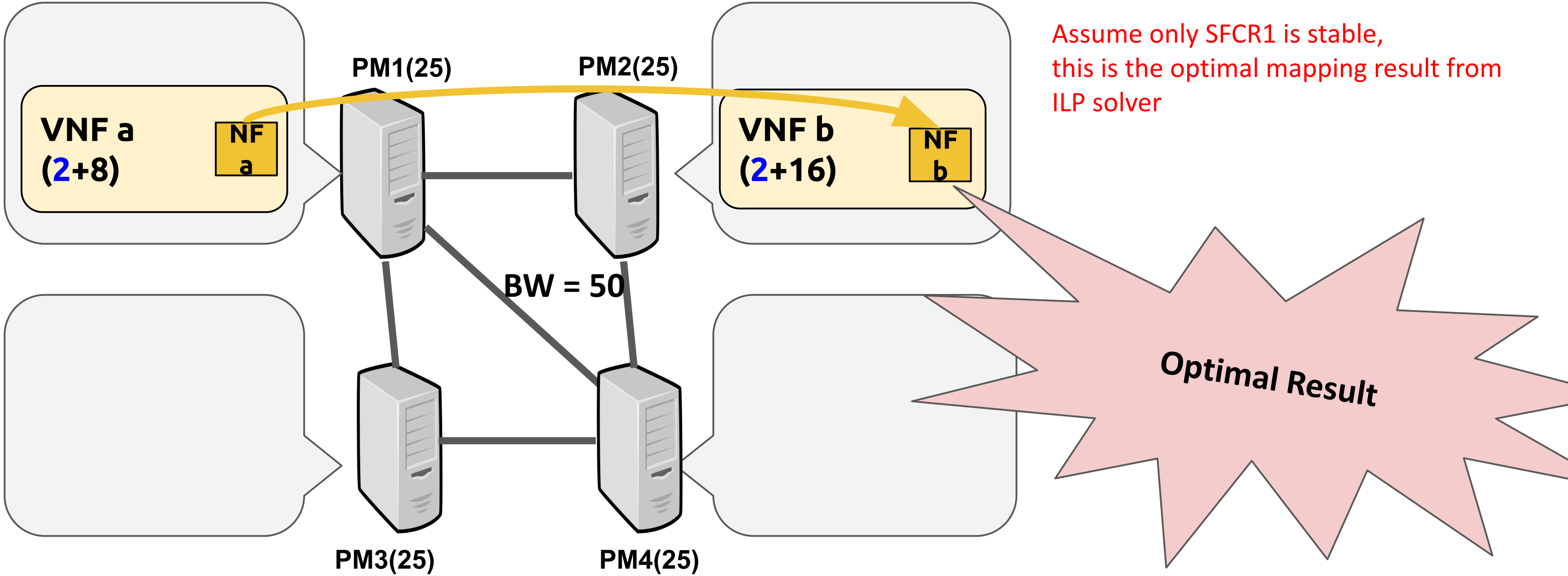
SFCR1 D=40

NF a (8) → NF b (16)

SFCR2 D=10

NF c (6) → NF d (8) → NF a (2) → NF e (10)

Basic resource consumption = 2 units



Assume only SFCR1 is stable,
this is the optimal mapping result from ILP solver

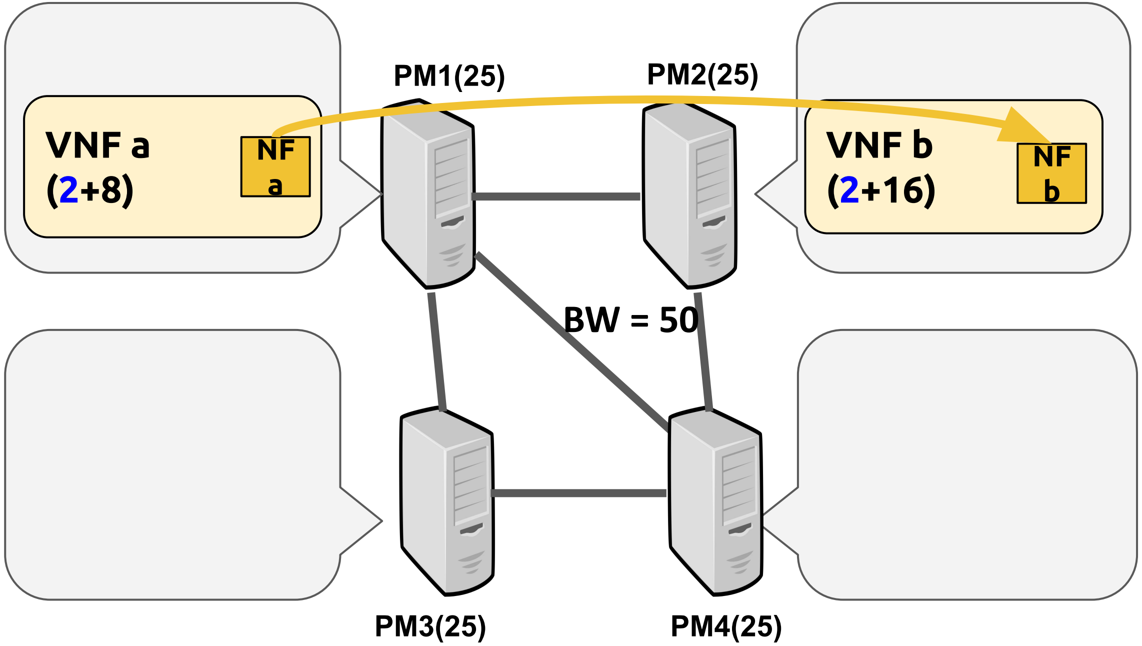
Greedy Placement Algorithm

- Three principles:
  - **Placement order:** SFCR with higher traffic is with higher priority
    - Contribute more revenue
    - Avoid resources fragmentation
  - **Allocation**: reuse deployed VNF
    - Reduce the basic resource consumption
    - Reduce the deployment cost
  - **Routing**: Place the NFs of a SFC close to each other
    - Minimize the end-to-end delay
    - Maximize the revenue of a SFC

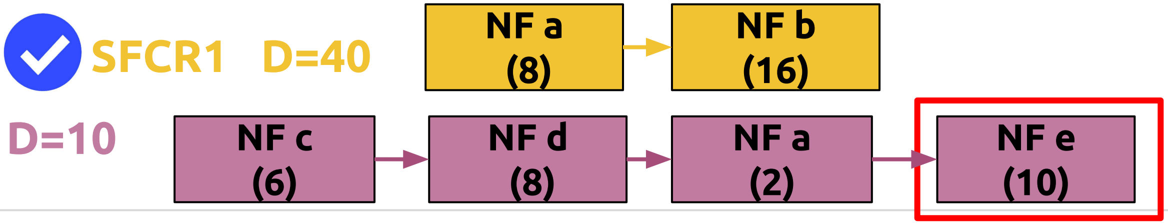
SFCR1 D=40: NF a (8) → NF b (16)

SFCR2 D=10: NF c (6) → NF d (8) → NF a (2) → NF e (10)

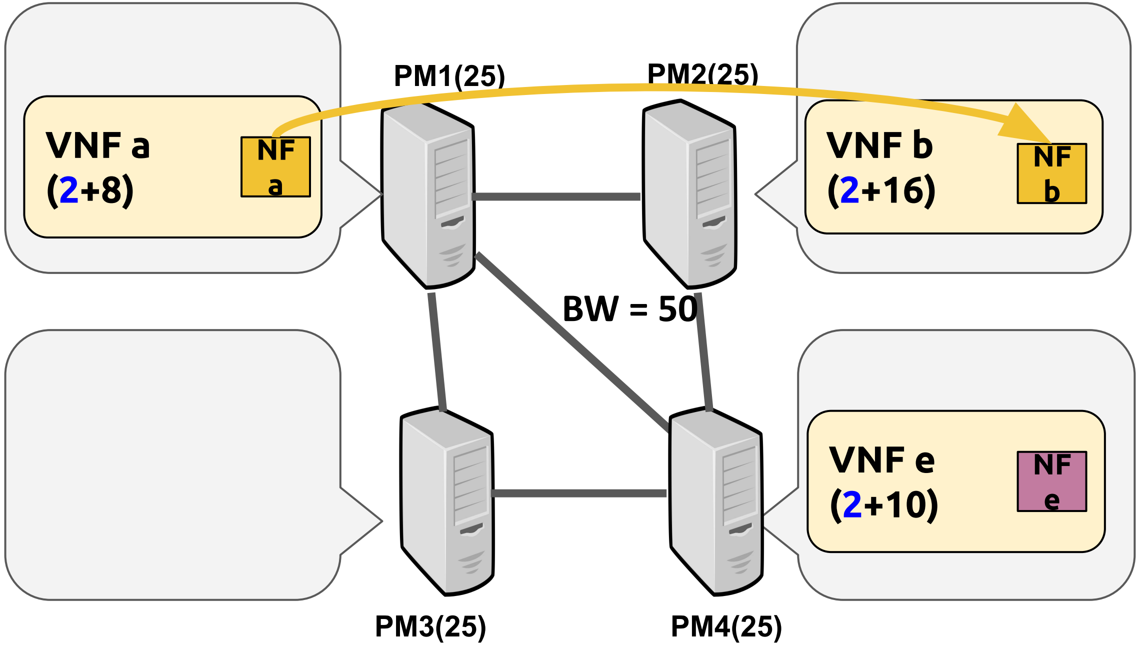
Basic resource consumption = 2 units



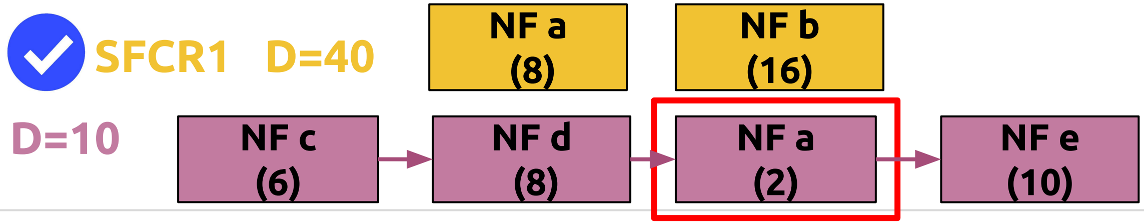
Step1: find a SFCR with the highest traffic demand



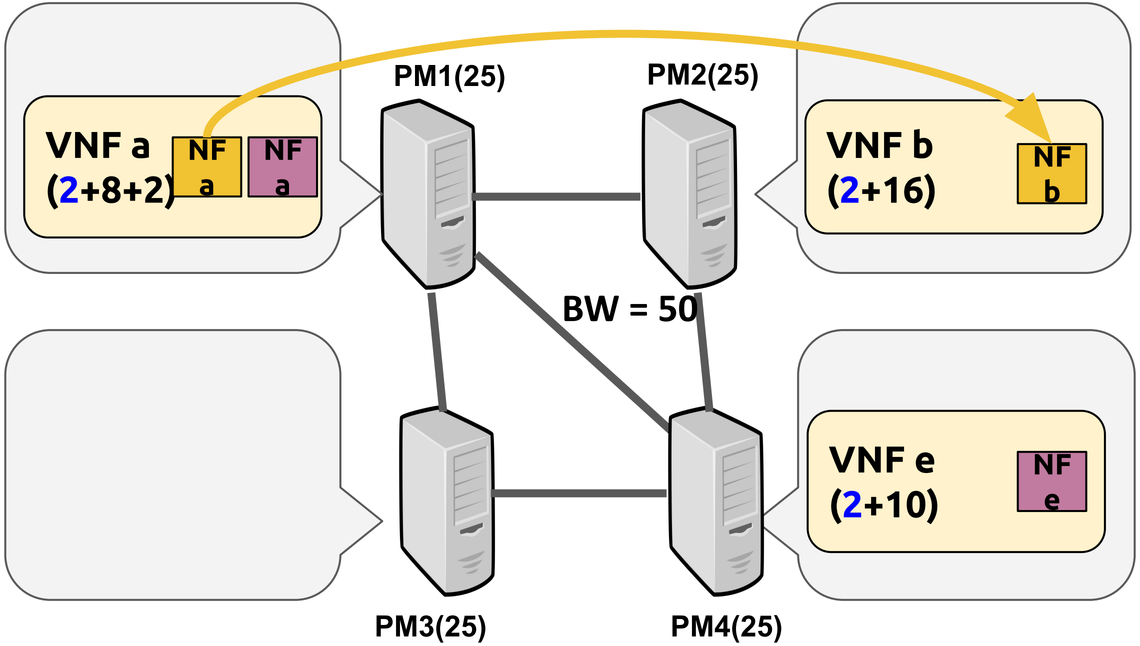
Basic resource consumption = 2 units



Step2: find a NF f in the SFCR2 with the highest node resource consumption, and place it on the node with the highest residual capacity.



Basic resource consumption = 2 units



Step3: based on the placement location of f, place its predecessor and successor NF ($fprev$, $fsucc$) according to the policy below.

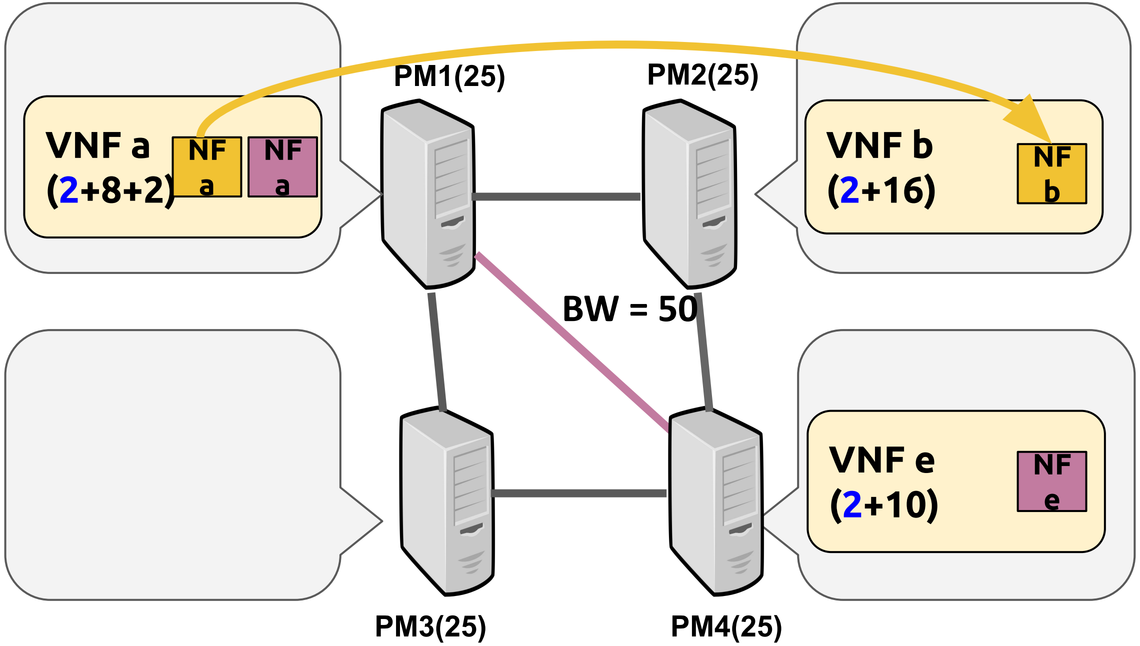
If the VNF of $fprev$ or $fsucc$ has been deployed on some nodes

- yes, place $fprev$ or $fsucc$ with the closest distance to f
- no, deploy a new VNF instance on the node which is closest distance to f

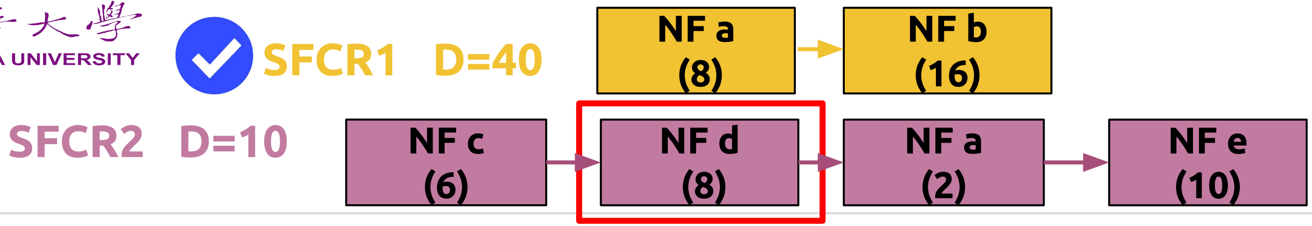
SFCR1 D=40: NF a (8) → NF b (16)

SFCR2 D=10: NF c (6) → NF d (8) → NF a (2) → NF e (10)

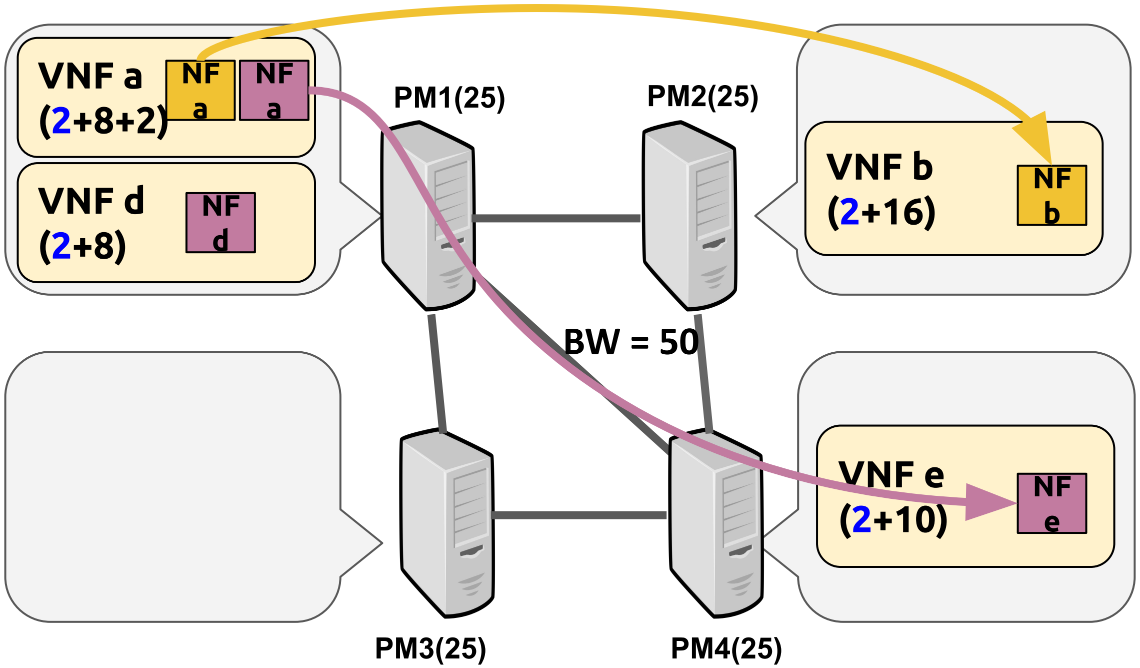
Basic resource consumption = 2 units



Step4: route the traffic from from *fprev* and *fsucc* to *f* through the shortest path with sufficient link capacity.



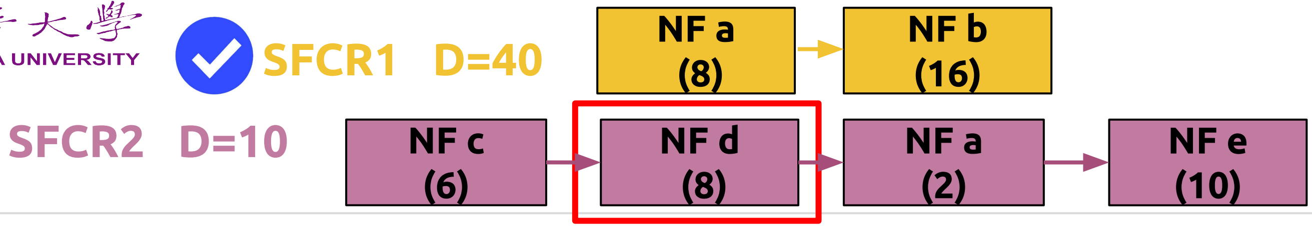
Basic resource consumption = 2 units



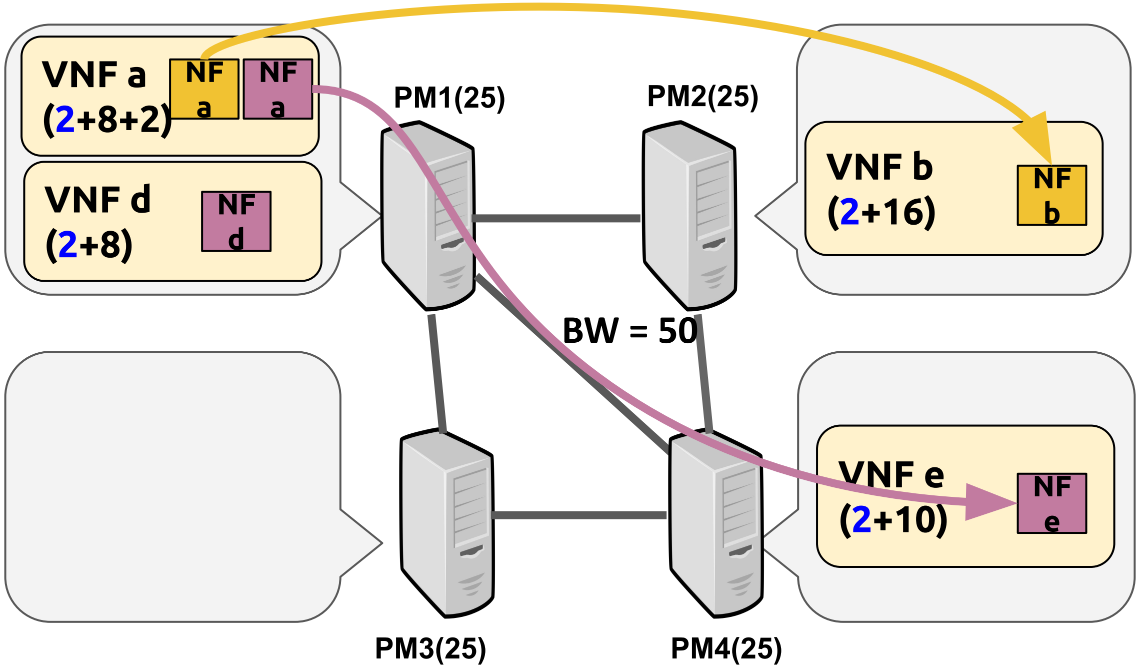
Step3: based on the placement location of f, place its predecessor and successor NF ($fprev$, $fsucc$) according to the policy below.

If the VNF of $fprev$ or $fsucc$ has been deployed on some nodes

- yes, place $fprev$ or $fsucc$ with the closest distance to f
- no, deploy a new VNF instance on the node which is closest distance to f



Basic resource consumption = 2 units

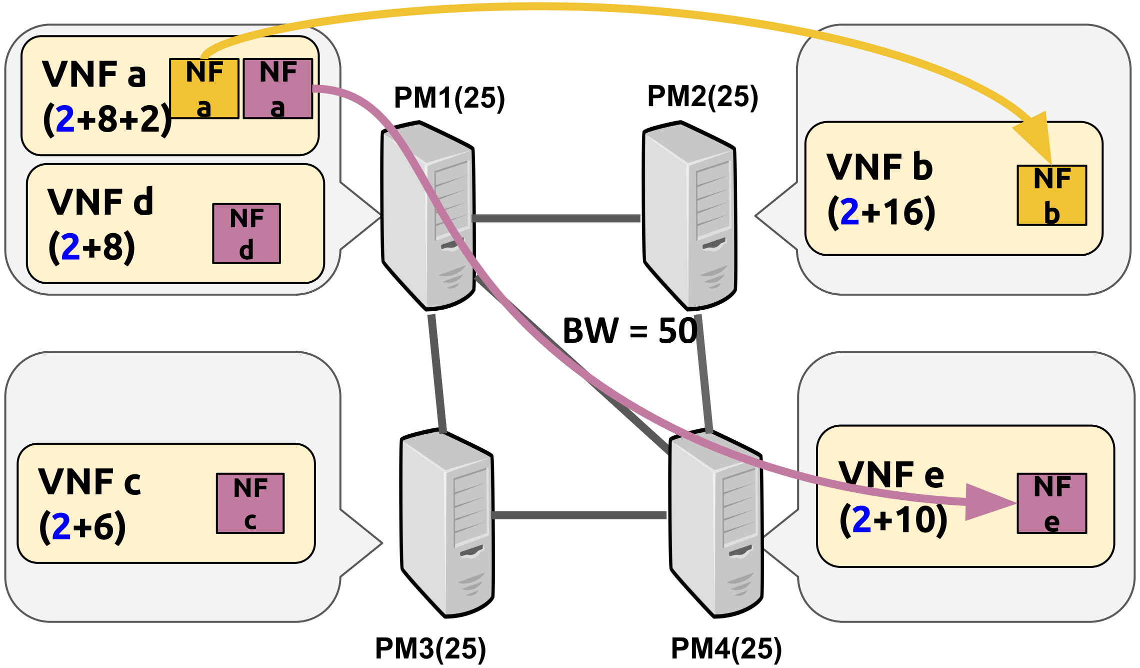


Step4: route the traffic from from *fprev* and *fsucc* to *f* through the shortest path with sufficient link capacity.

SFCR1 D=40: NF a (8) → NF b (16)

SFCR2 D=10: NF c (6) → NF d (8) → NF a (2) → NF e (10)

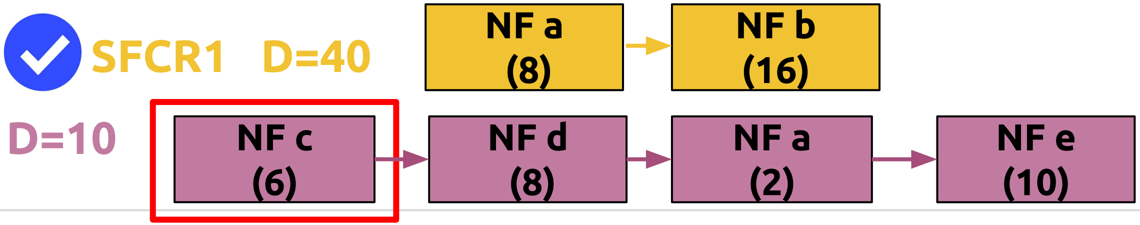
Basic resource consumption = 2 units



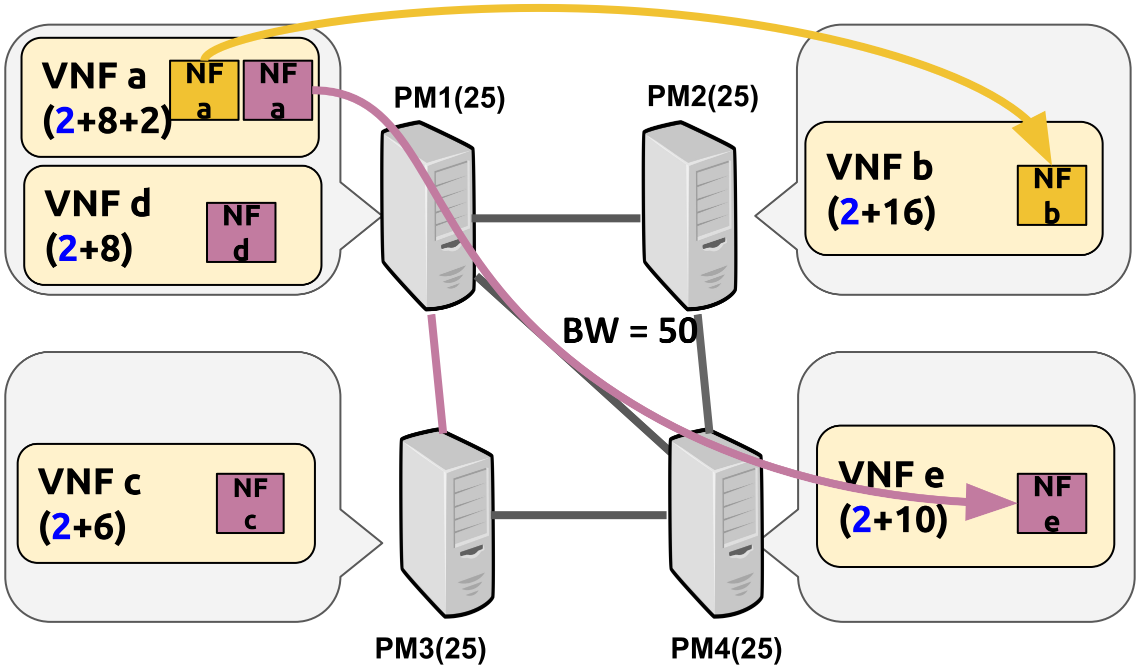
Step3: based on the placement location of f, place its predecessor and successor NF ($fprev$, $fsucc$) according to the policy below.

If the VNF of $fprev$ or $fsucc$ has been deployed on some nodes

- yes, place $fprev$ or $fsucc$ with the closest distance to f
- no, deploy a new VNF instance on the node which is closest distance to f



Basic resource consumption = 2 units

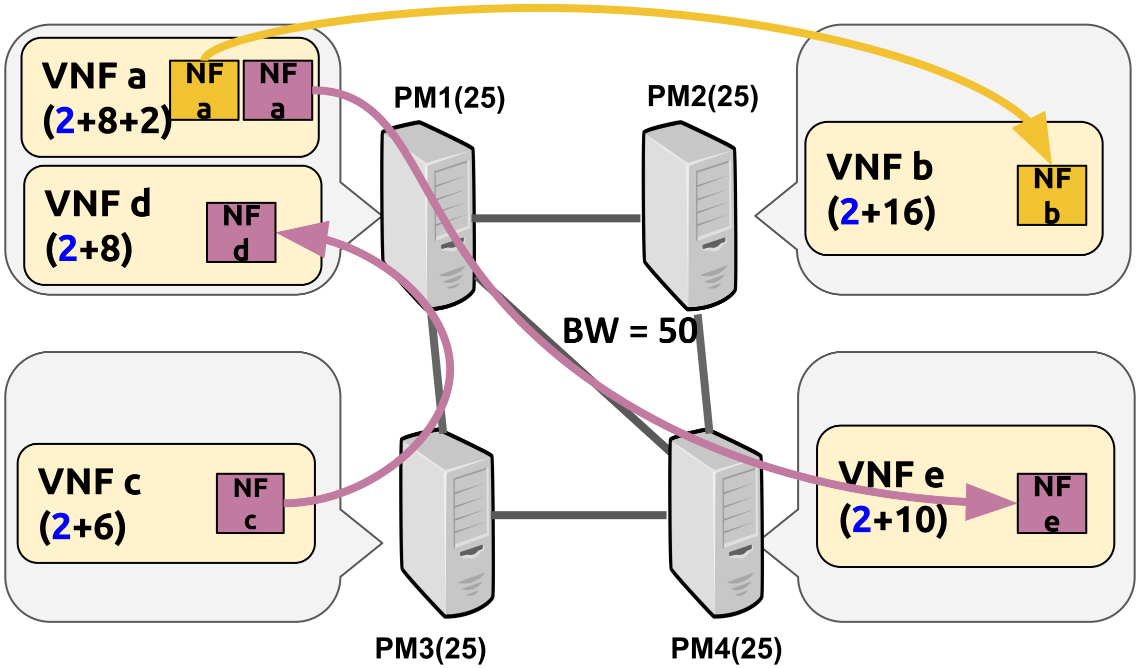


Step4: route the traffic from from *fprev* and *fsucc* to *f* through the shortest path with sufficient link capacity.

SFCR1 D=40 — NF a (8) → NF b (16)

SFCR2 D=10 — NF c (6) → NF d (8) → NF a (2) → NF e (10)

Basic resource consumption = 2 units



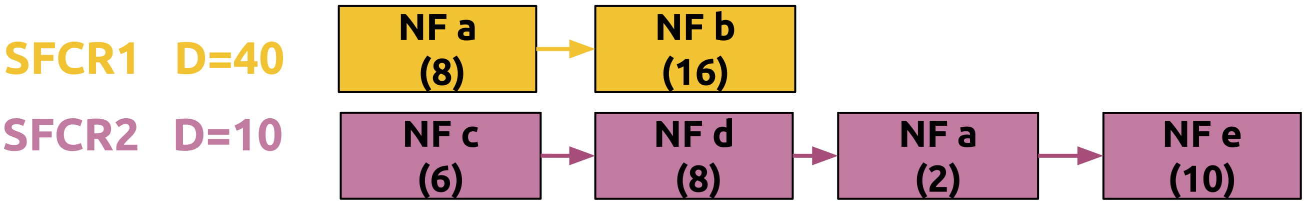
Step4: route the traffic from from *fprev* and *fsucc* to *f* through the shortest path with sufficient link capacity.

Greedy Reconfiguration Algorithm

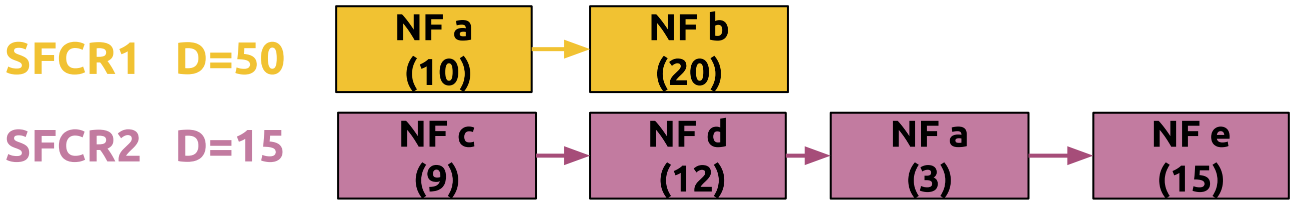
- Dynamic traffic
- Reconfiguration is triggered when a violation is occurred on a node.

Dynamic Traffic

Time: t

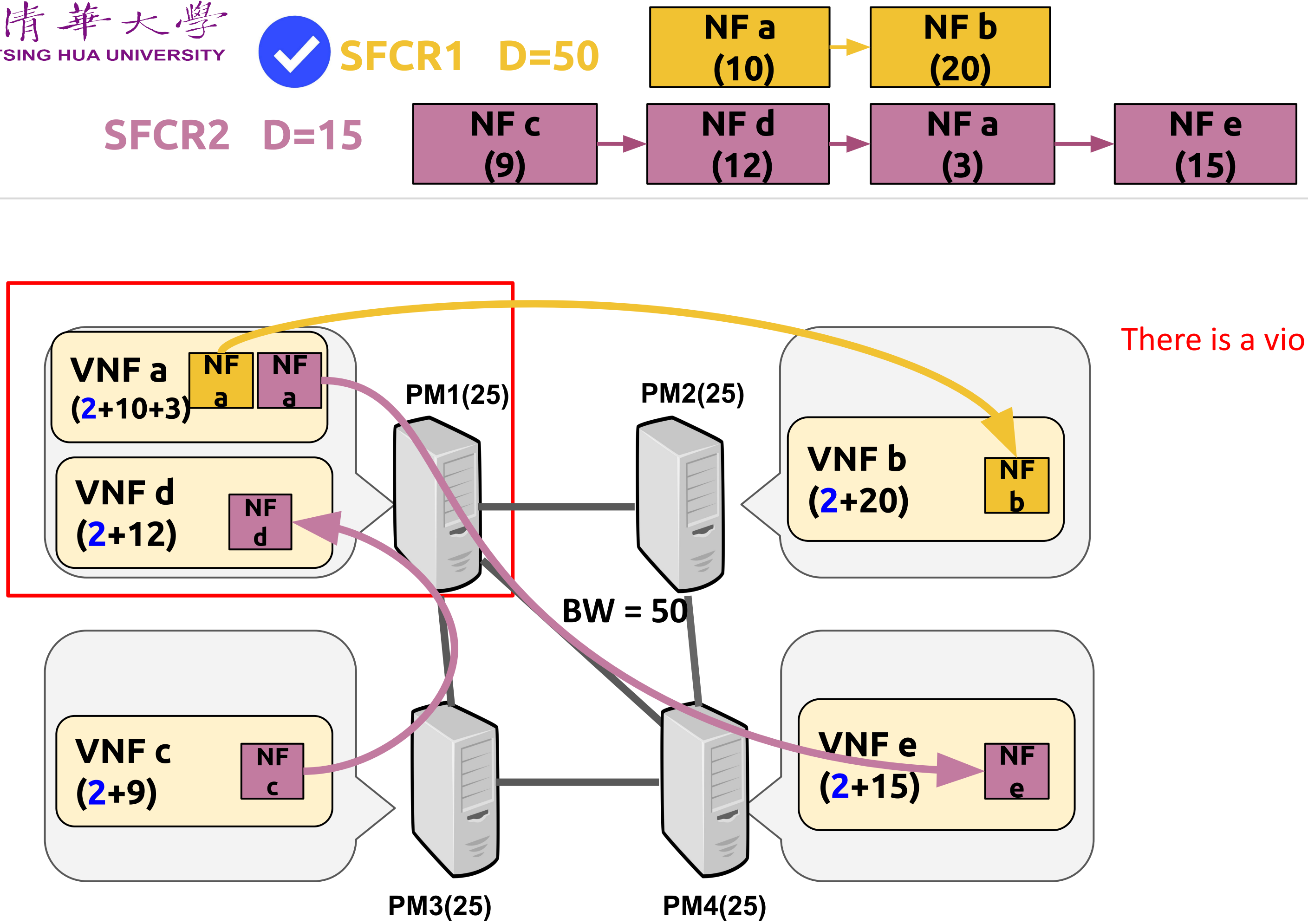


Time: t + 1



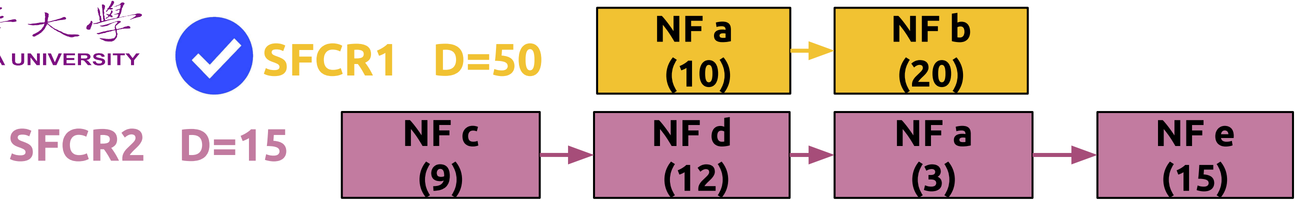
SFCR1 D=50

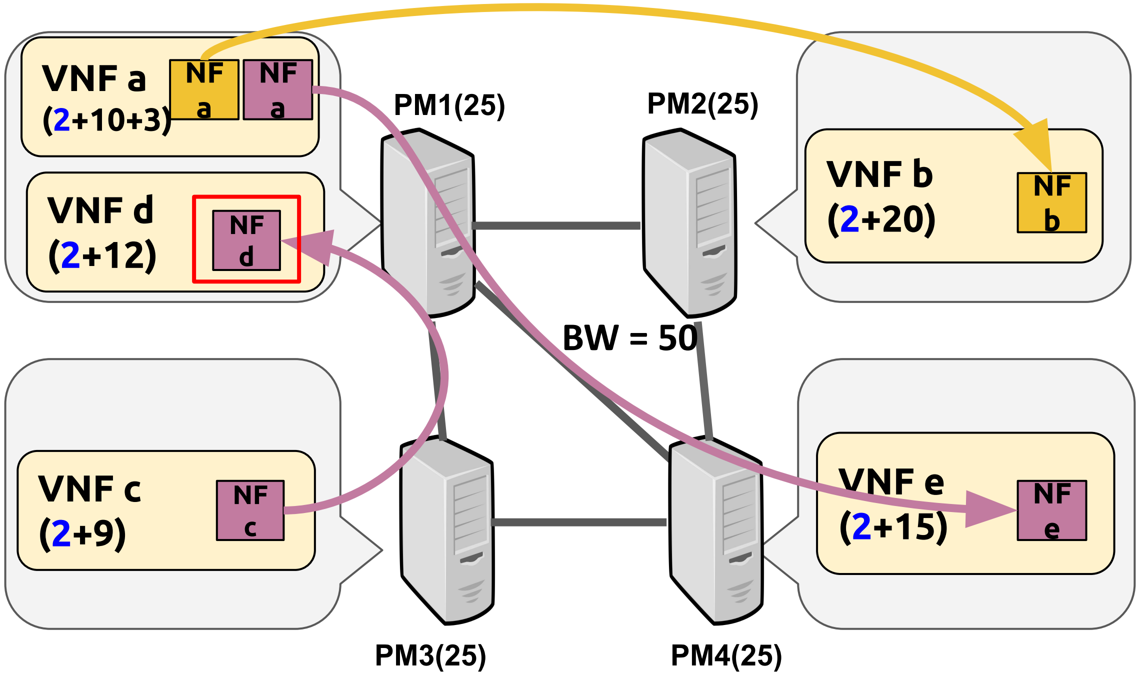
SFCR2 D=15



There is a violation on PM1 !!!

國立清華大學
NATIONAL TSING HUA UNIVERSITY

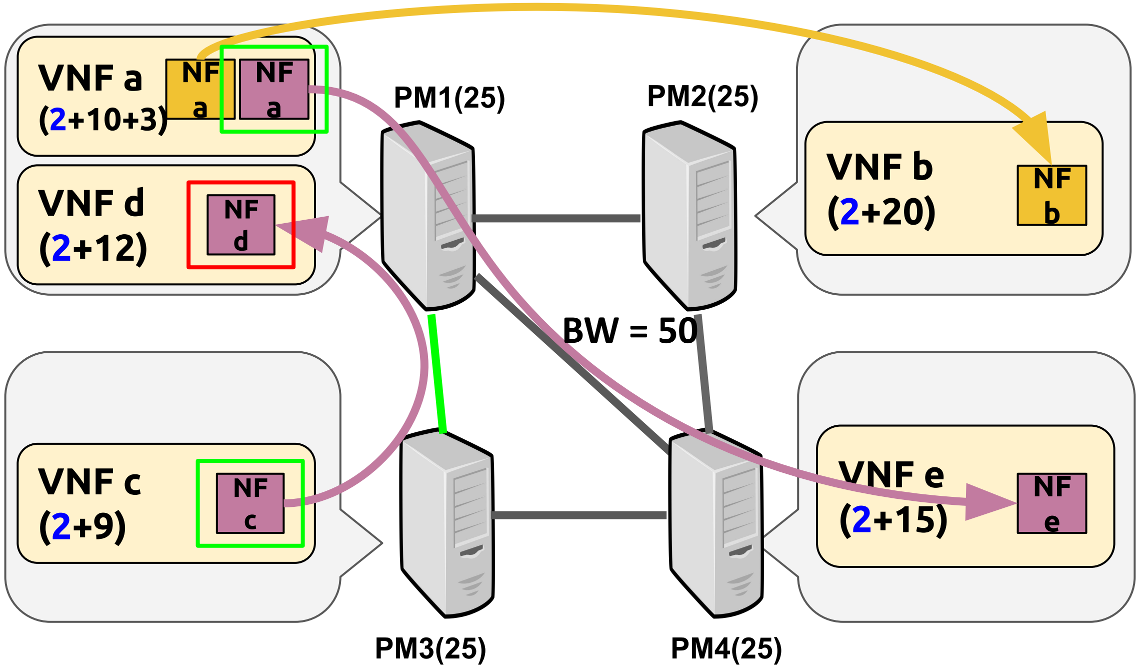




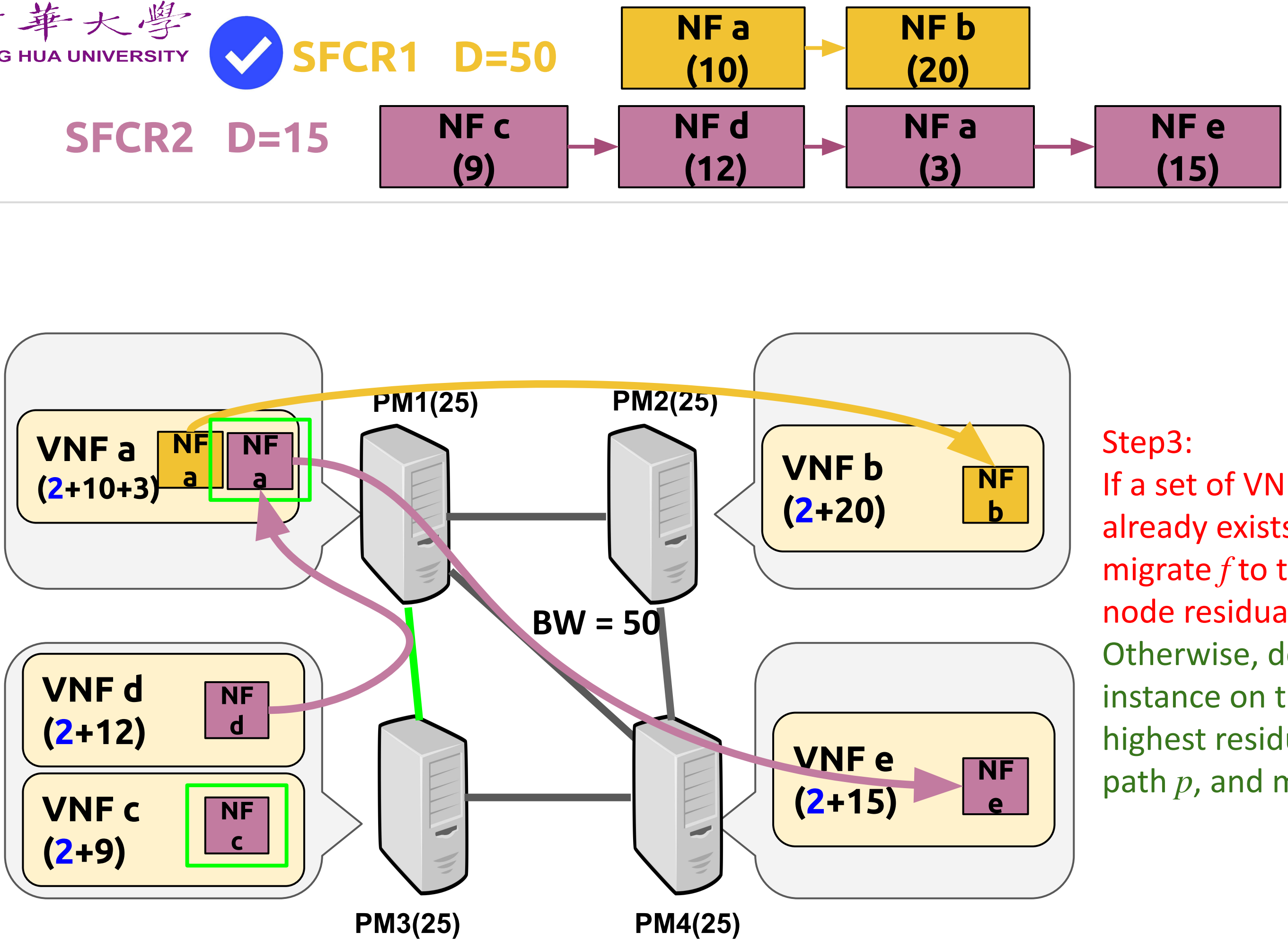
Step1: find the NF f with the highest resource consumption.

SFCR1 D=50: NF a (10) → NF b (20)

SFCR2 D=15: NF c (9) → NF d (12) → NF a (3) → NF e (15)



Step2: find the shortest path p between the predecessor and successor NFs of f.



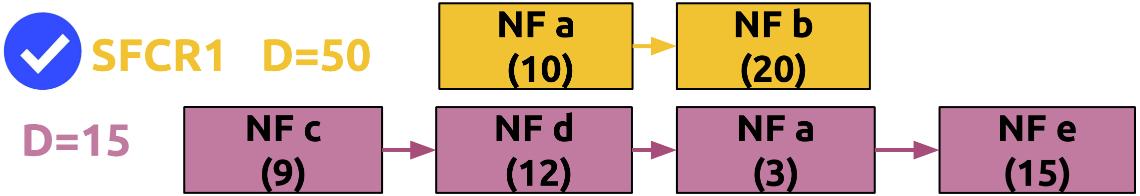
Step3:

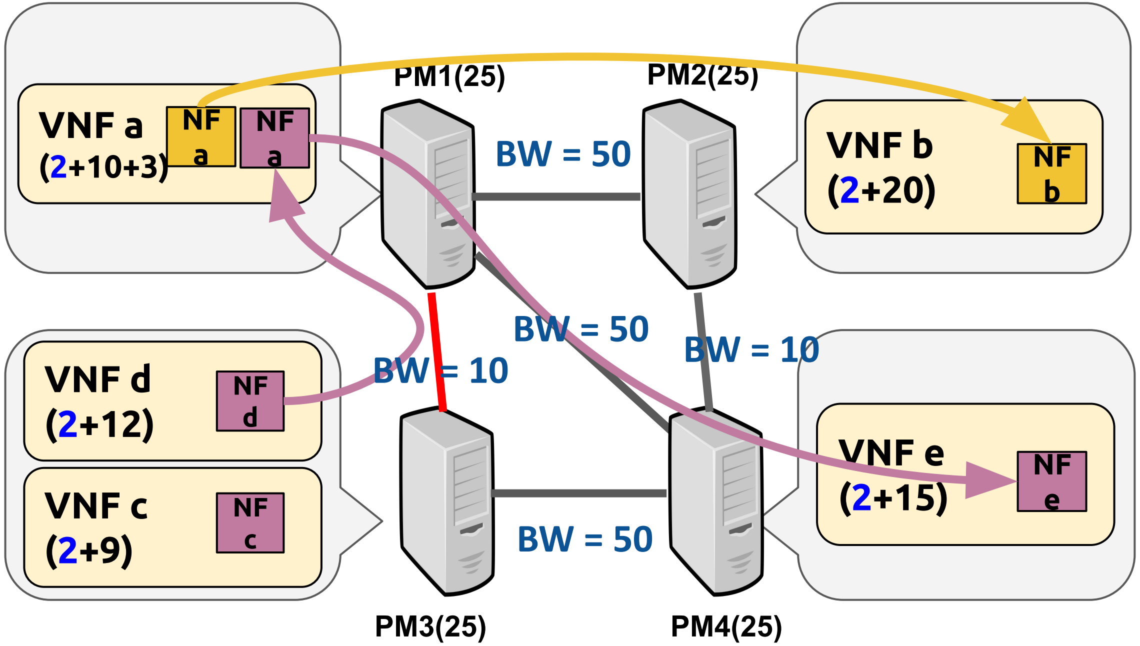
If a set of VNFs that can serve f already exists along the path, migrate f to the VNF with the highest node residual capacity.

Otherwise, deploy a new VNF instance on the node with the highest residual capacity along the path p, and migrate f to the VNF.

Greedy Reconfiguration Algorithm

- Reconfiguration is triggered when a violation is occurred on a link.

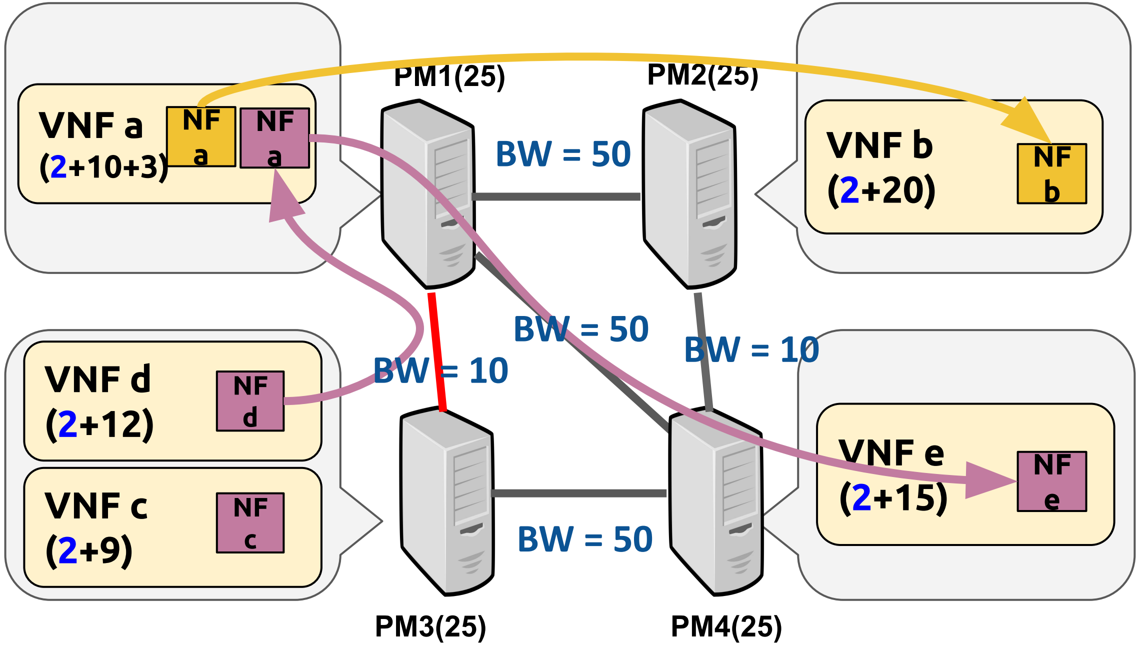




There is a violation on the link between PM1 and PM3 !!!

SFCR1 D=50: NF a (10) → NF b (20)

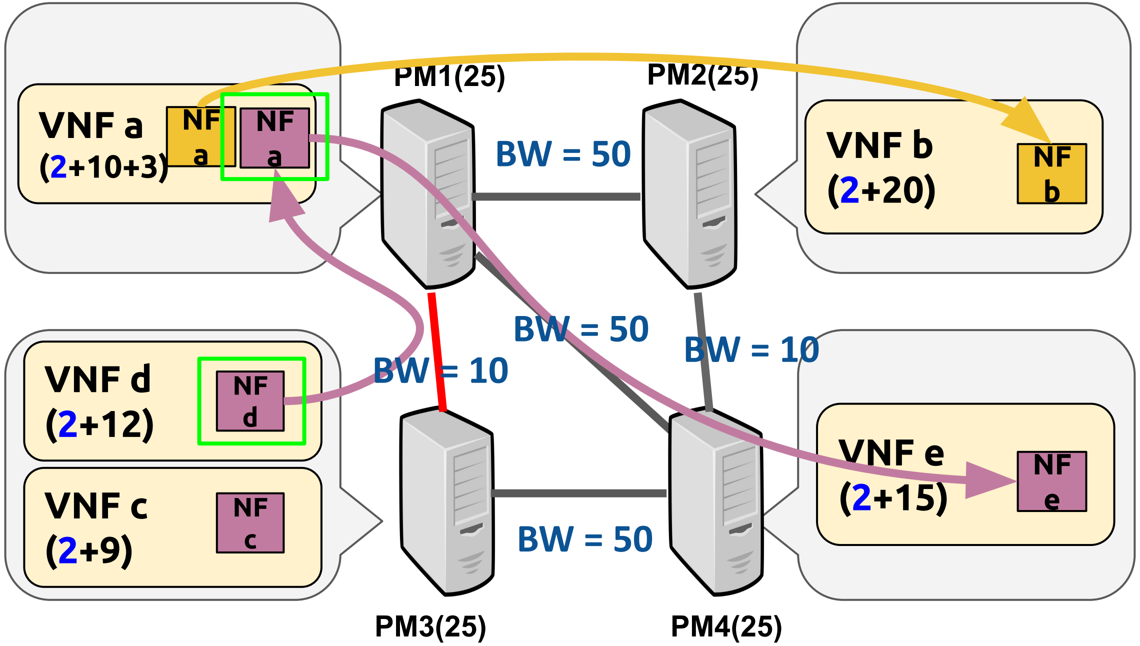
SFCR2 D=15: NF c (9) → NF d (12) → NF a (3) → NF e (15)



Step1: from the violating link l, find the SFCR s with the least traffic demand.

SFCR1 D=50: NF a (10) → NF b (20)

SFCR2 D=15: NF c (9) → NF d (12) → NF a (3) → NF e (15)



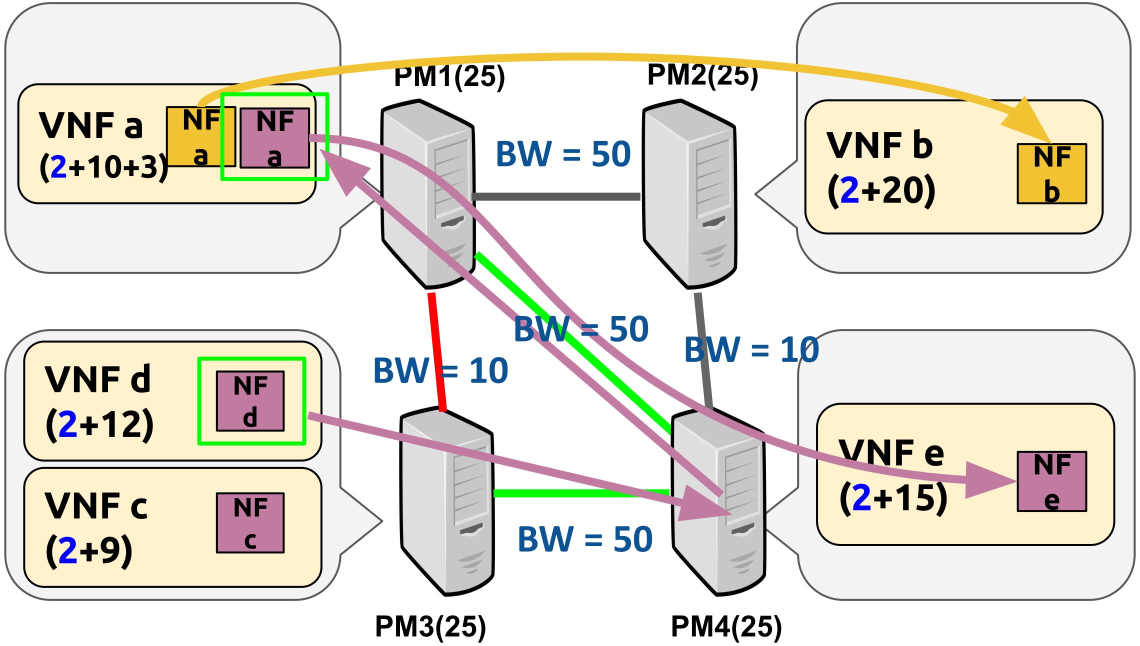
Step2: find the NF fi in s that uses the link l to route the traffic to its next NF fi+1 in s.

SFCR1 D=50

NF a (10) → NF b (20)

SFCR2 D=15

NF c (9) → NF d (12) → NF a (3) → NF e (15)



Step3: reroute the traffic from f_i to f_i+1 through the shortest path between their node locations without using link l .

Outline

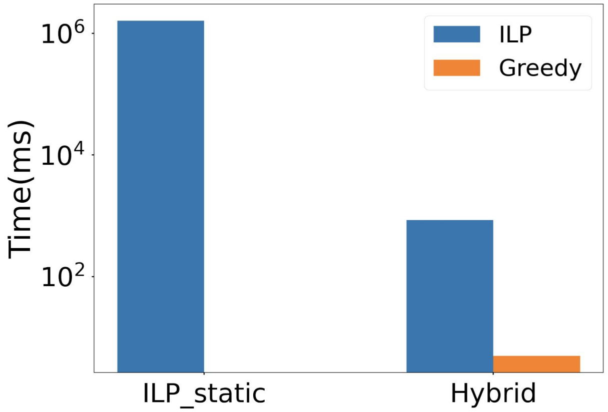
- Motivation & Objective
  - Introduction of Network Function Virtualization
  - Placement Problems and Challenges
  - Existing Approaches
- Methodology
- Experimental Evaluations
- Conclusion

Experimental Setup

- Physical graph: 6 nodes and 25 links
- SFCRs:
  - 6 SFCRs with varied length and latency threshold
  - Workload: a sinusoidal signal + random value
  - Variance of the traffic demand
    - Stable SFCRs is lower than 1
    - Unstable SFCRs is over than 4
- Comparison Method
  - ILP_static :
    - At the first time interval: use ILP solver to decide
    - At the remaining time interval: if there is a violation, SFCRs are rejected

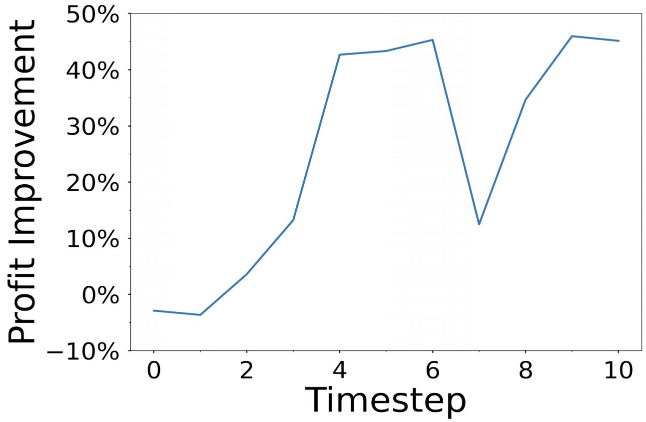
Comparison of Computation Time

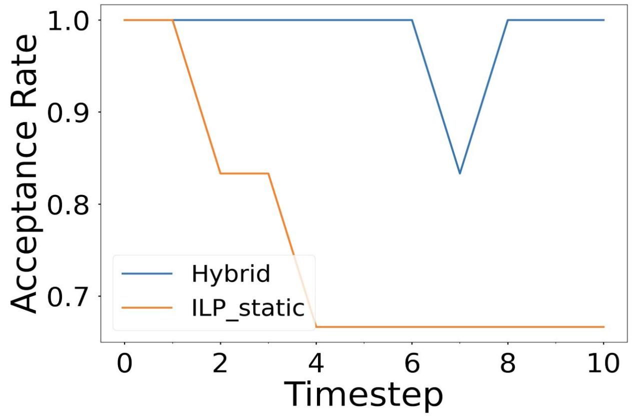
- Show the first time interval
  - Time : ILP_static > Hybrid
- Results
  - Our approach can always make timing decisions
  - Greater improvements can be expected when considering larger networks and more SFCRs in the problem



Comparison of Profit

- At the first time interval
  - Hybrid achieves a similar result as ILP_static
- At the remaining time interval
  - Hybrid gets even higher profit than ILP_static
  - Up to 45% profit improvement at one time interval and 24% improvement across all time intervals





Outline

- Motivation & Objective
  - Introduction of Network Function Virtualization
  - Placement Problems and Challenges
  - Existing Approaches
- Methodology
- Experimental Evaluations
- Conclusion

Conclusions

- Solving VNF placement
  - Using an ILP solver is time-consuming, but has an optimal mapping result
  - Using a greedy algorithm is fast, but only has an approximated mapping result.
- We propose a hybrid VNF placement approach to maximize the net-profit of a network service provider
  - ILP + a greedy placement strategy
    - Overcome the time complexity problem of ILP solutions under time-varied workload
  - A greedy reconfiguration strategy
    - Resolve resource violations caused by time-varied workload
- Our hybrid method can get up to 45% improvement at a given time interval, and overall 24% improvement over all the time intervals