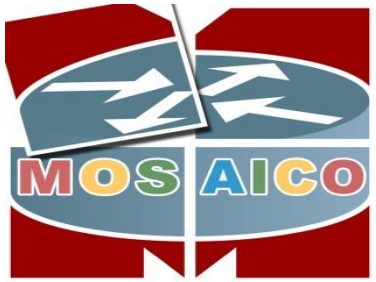


Placement et Chainage de Fonction de Services décomposées en Micro-services : Modélisation et Résolution

Hichem MAGNOUCHE* – Guillaume DOYEN – Caroline PRODHON***

*Université de Technologie de Troyes, France

**IMT Atlantique-Rennes, France

NetOpt 2022

10/11/2022 – Paris

Sommaire

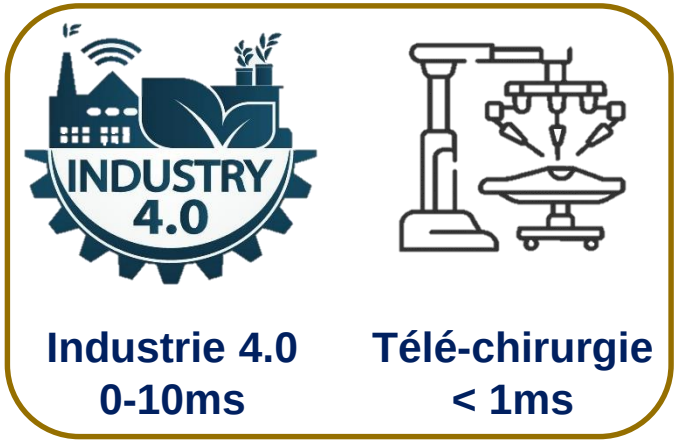
- Services à très faible latence et approche micro-services
- Modélisation de l'approche de placement et de chaînage des micro-services
- Vers un placement équilibré des services à très faible latence et de services Best Effort
- Approche de résolution heuristique
- Conclusion et travaux futurs

Sommaire

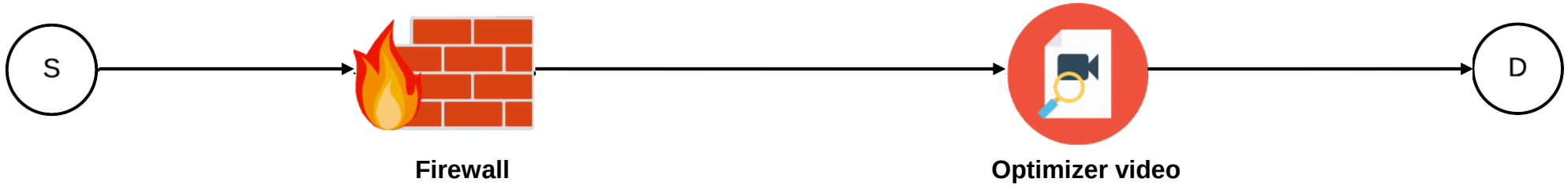
- Services à très faible latence et approche micro-services
- Modélisation de l'approche de placement et de chaînage des micro-services
- Vers un placement équilibré des services à très faible latence et de services Best Effort
- Approche de résolution heuristique
- Conclusion et travaux futurs

Services à très faible latence et approche micro-services

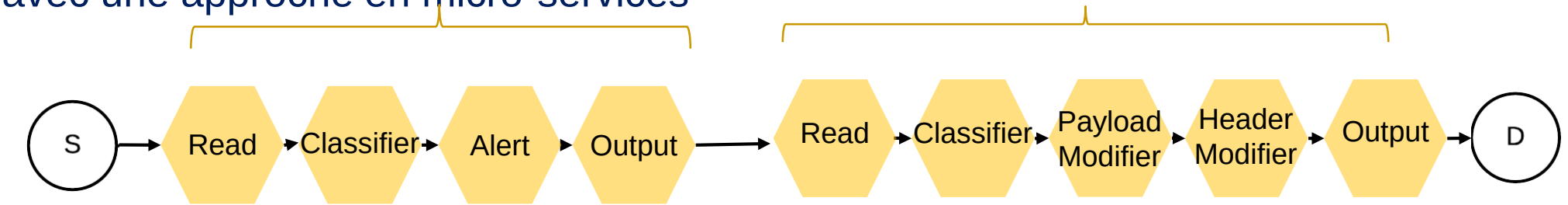
- Utilisation croissante de services à très faible latence
- Temps de transit limité dans l'infrastructure du réseau
- Différentes façons de réduire la latence
  - Réduire les délais d'attente (ex. TCP Prague et Dual PI2 dans L4S)
  - **Optimisation du chaînage et du placement des fonctions réseau**
  - **Décomposition de VNF en micro-services**



- SFC avec une approche de VNF monolithiques

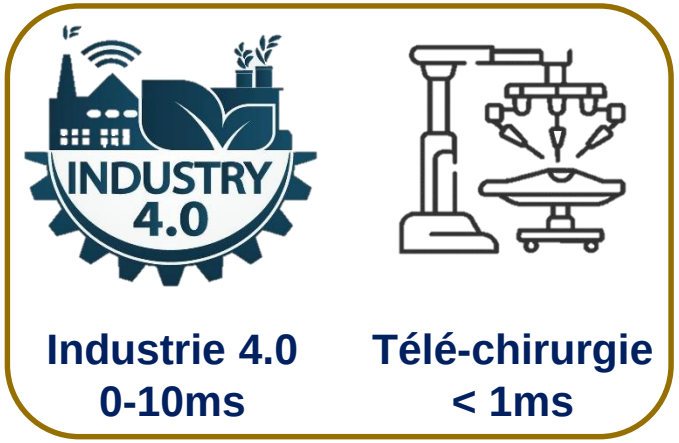


- SFC avec une approche en micro-services

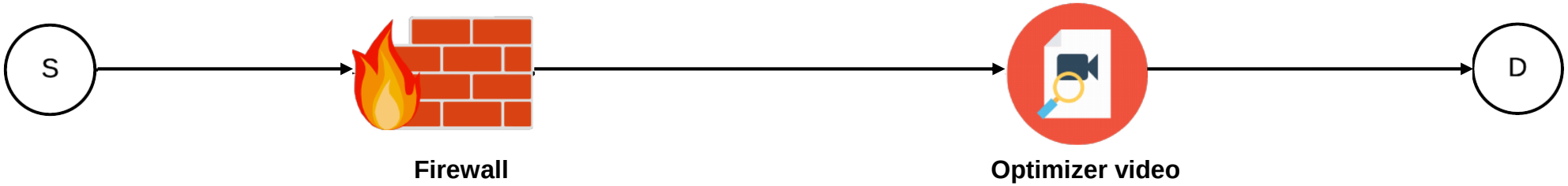


Services à très faible latence et approche micro-services

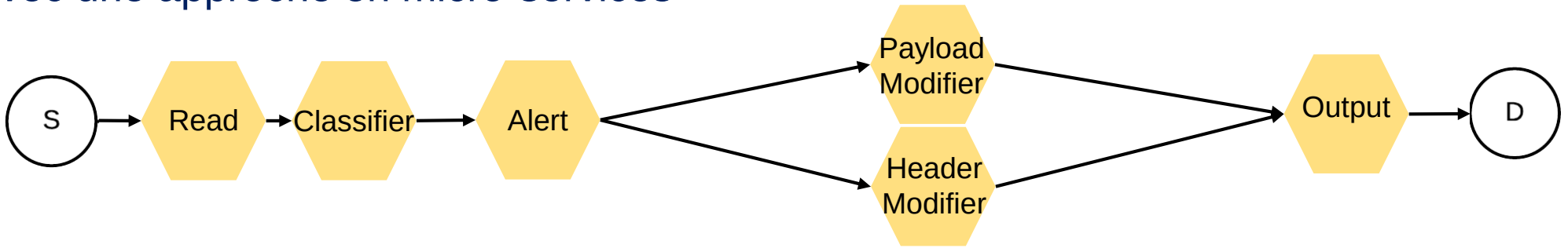
- Utilisation croissante de services à très faible latence
- Temps de transit limité dans l'infrastructure du réseau
- Différentes façons de réduire la latence
  - Réduire les délais d'attente (ex. TCP Prague et Dual PI2 dans L4S)
  - **Optimisation du chaînage et du placement des fonctions réseau**
  - **Décomposition de VNF en micro-services**



- SFC avec une approche de VNF monolithiques



- SFC avec une approche en micro-services



Décomposition en micro-services

- Bénéfices
  - Réduction de la longueur totale des SFC : mutualisation et parallélisation
  - Agilité dans le déploiement
  - Simplicité de développement, de mise à jour et de maintenance

- Verrous
  - Comment optimiser le déploiement de SFC dans le cadre d'une approche micro-services ?
    - Mutualisation
      - Quand faut-il mutualiser des micro-services ?
    - Parallélisation
      - Faut-il paralléliser les micro-services sur le même nœud ? Sur des nœuds différents ?
    - Restructuration de la SFC (Service Function Chain)
      - La restructuration des SFC doit-elle être optimisée ou est-elle statique ?

Mutualisation et parallélisation

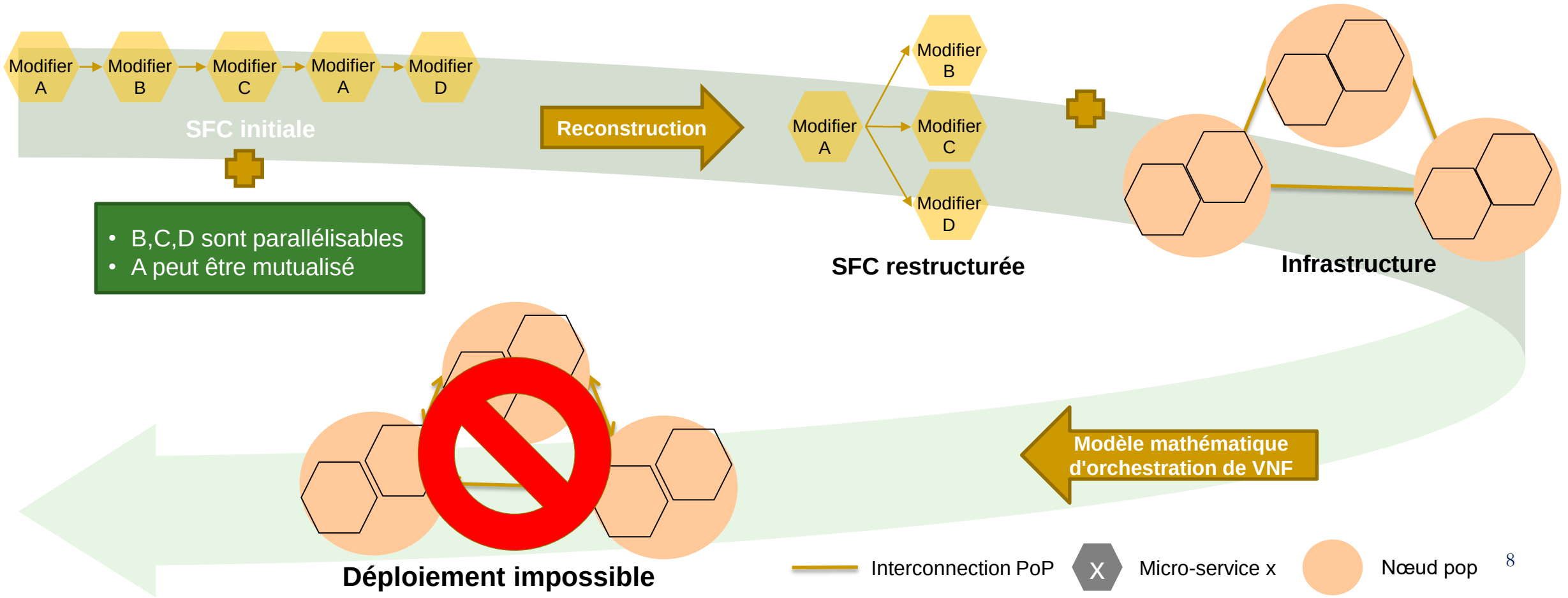
- Mutualisation
  - Nécessite une analyse des traitements de paquets effectués entre les micro-services à mutualiser
  - [1][2] réalisent cette analyse et proposent une table de mutualisation.
  - La mutualisation est faite en amont du déploiement

- Parallélisation
  - L'objectif est la réduction du temps d'exécution
  - Déjà exploité dans l'orchestration monolithique de VNF [3][4]
    - Externe [3] : paralléliser des VNF sur différents nœuds
    - Interne [4] : paralléliser uniquement des VNF sur le même nœud en utilisant la mémoire partagée (par exemple, DPDK)

[1] Z. Meng, J. Bi, H. Wang, C. Sun and H. Hu, "MicroNF: An Efficient Framework for Enabling Modularized Service Chains in NFV," in IEEE Journal on Selected Areas in Communications, Aug. 2019
[2] S. R. Chowdhury, Anthony, H. Bian, T. Bai and R. Boutaba, "A Disaggregated Packet Processing Architecture for Network Function Virtualization" in IEEE Journal on Selected Areas in Communications, vol. 38, no. 6, pp. 1075-1088, June 2020
[3] Yang Zhang, Bilal Anwer, Vijay Gopalakrishnan, Bo Han, Joshua Reich, Aman Shaikh, and Zhi-Li Zhang. 2017. ParaBox: Exploiting Parallelism for Virtual Network Functions in Service Chaining. In Proceedings of the Symposium on SDN Research (SOSR '17). Association for Computing Machinery, New York, NY, USA.
[4] Chen Sun, Jun Bi, Zhilong Zheng, Heng Yu, and Hongxin Hu. 2017. NFP: Enabling Network Function Parallelism in NFV. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 43–56.

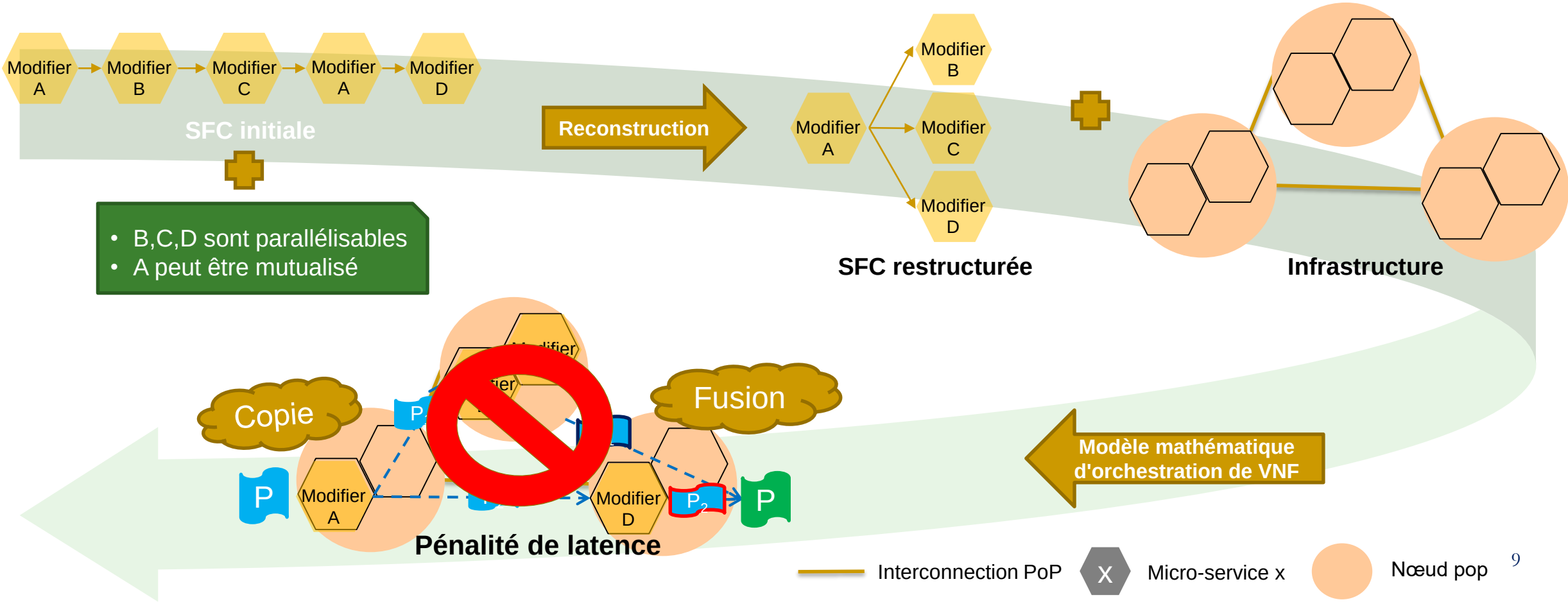
Limites de l'état de l'art

- Les modèles monolithiques ne profitent pas des caractéristiques des micro-services
- La restructuration des SFC avant le déploiement n'est pas toujours optimale
  - Situation de non-placement dans certaines configurations avec parallélisation **interne** uniquement



Limites de l'état de l'art

- Les modèles monolithiques ne profitent pas des caractéristiques des micro-services
- La restructuration des SFC avant le déploiement n'est pas toujours optimale
  - Détérioration de la latence dans certains configurations avec parallélisation **interne** et **externe**



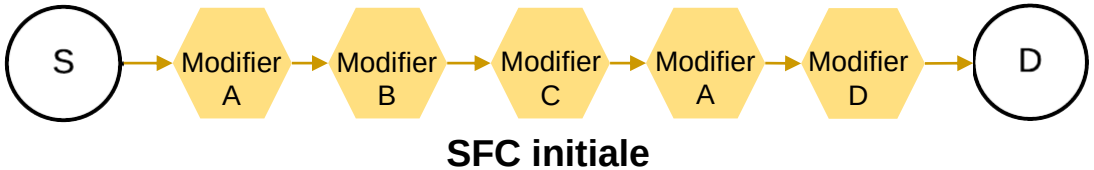
Sommaire

- Services à très faible latence et approche micro-services
- Modélisation de l'approche de placement et de chaînage des micro-services
- Vers un placement équilibré des services à très faible latence et de services Best Effort
- Approche de résolution heuristique
- Conclusion et travaux futurs

Approche développée de placement et de chaînage de micro-services

- Une approche en deux étapes[5]
  - Algorithme de prétraitement
  - MILP (Mixed Integer Linear Program) de placement et de chaînage

- Étape 1 : algorithme de prétraitement
  - Mutualiser les micro-services lorsque cela est possible
  - Identifier les micro-services parallélisables
    - Si deux micro-services peuvent être parallélisés, ils sont retirés de la liste de précédence de chacun d'eux



SFC initiale

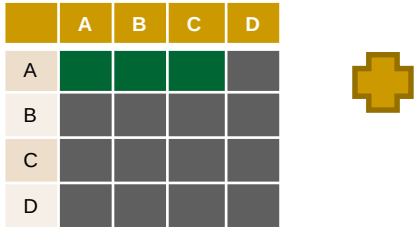


Tableau de mutualisation

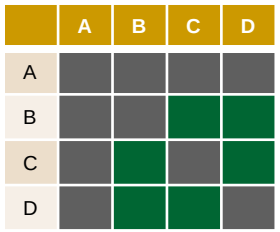
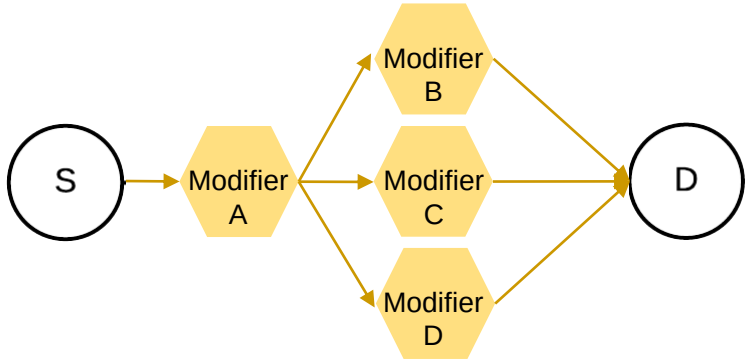


Tableau de parallélisation

Algorithme de prétraitement



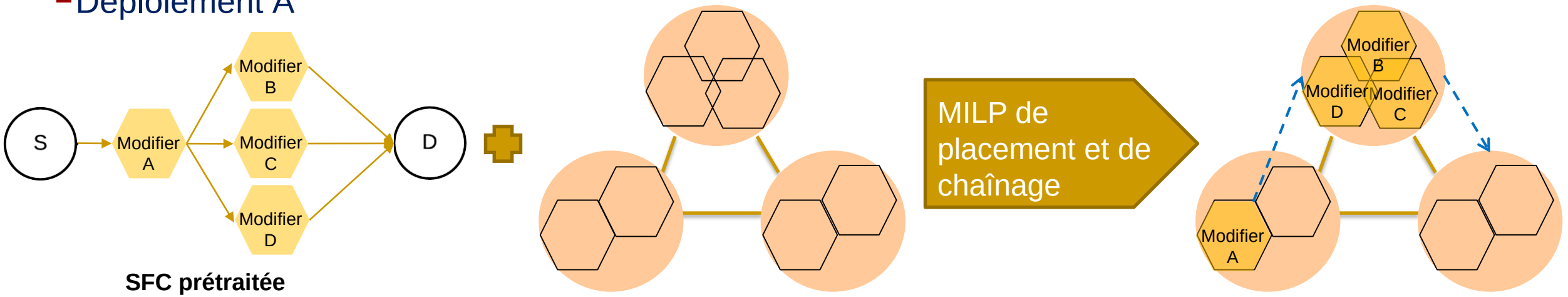
SFC prétraitée

[5]H. Magnouche, G. Doyen and C. Prodhon, "Leveraging Micro-Services for Ultra-Low Latency: An optimization Model for Service Function Chains Placement," 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), 2022, pp. 198-206, doi: 10.1109/NetSoft54395.2022.9844040.

Approche développée de placement et de chaînage de micro-services

- Étape 2 : MILP* de placement et de chaînage
  - Fonction objectif : minimiser l'écart entre la latence prescrite et la latence effective
  - Familles principales de contraintes
    - Ordre des micro-services
    - Respect de la mémoire disponible et de la capacité de débit sur les nœuds et les arcs
  - Optimiser la parallélisation des micro-services en fonction de la configuration de l'infrastructure réseau

- Déploiement A

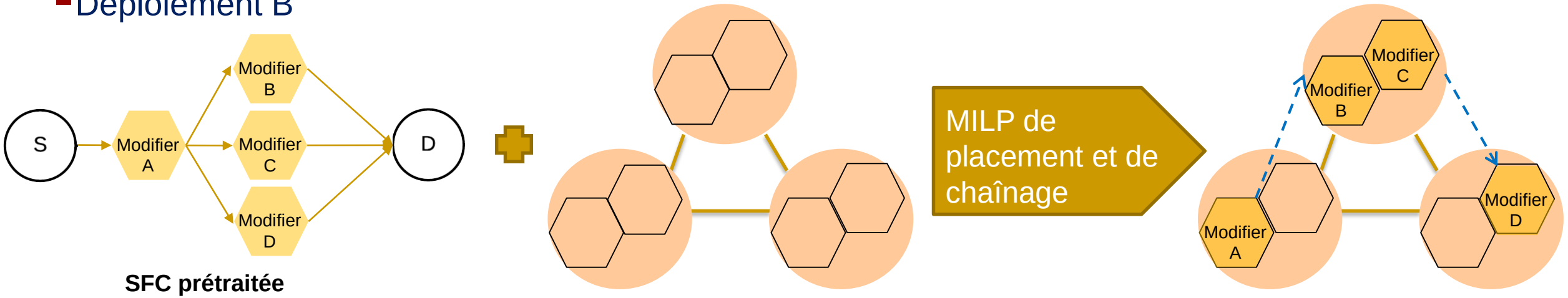


*MILP: Mixed Integer Linear Program

Approche développée de placement et de chaînage de micro-services

- Étape 2 : MILP* de placement et de chaînage
  - Fonction objectif : minimiser l'écart entre la latence prescrite et la latence effective
  - Familles principales de contraintes
    - Ordre des micro-services
    - Respect de la mémoire disponible et de la capacité de débit sur les nœuds et les arcs
  - Optimiser la parallélisation des micro-services en fonction de la configuration de l'infrastructure réseau

- Déploiement B

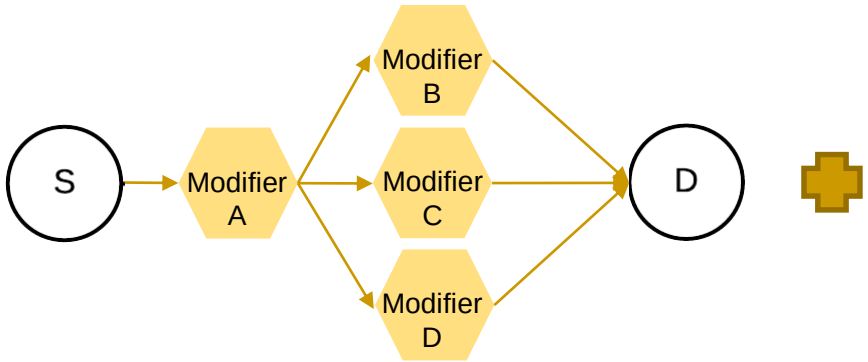


*MILP: Mixed Integer Linear Program

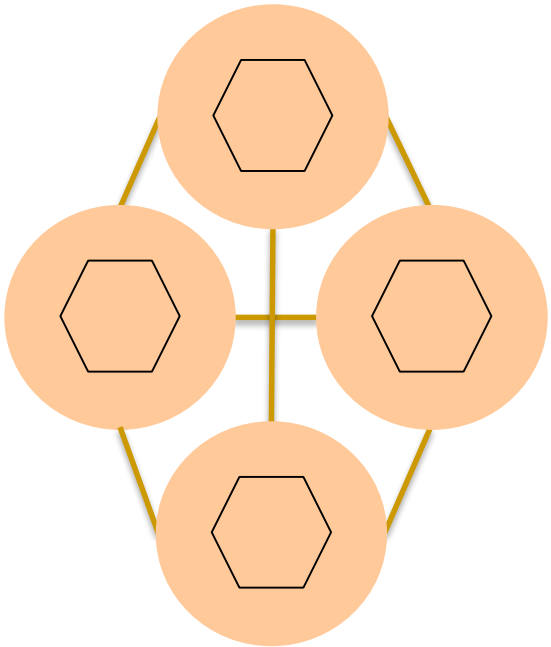
Approche développée de placement et de chaînage de micro-services

- Étape 2 : MILP* de placement et de chaînage
  - Fonction objectif : minimiser l'écart entre la latence prescrite et la latence effective
  - Familles principales de contraintes
    - Ordre des micro-services
    - Respect de la mémoire disponible et de la capacité de débit sur les nœuds et les arcs
  - Optimiser la parallélisation des micro-services en fonction de la configuration de l'infrastructure réseau

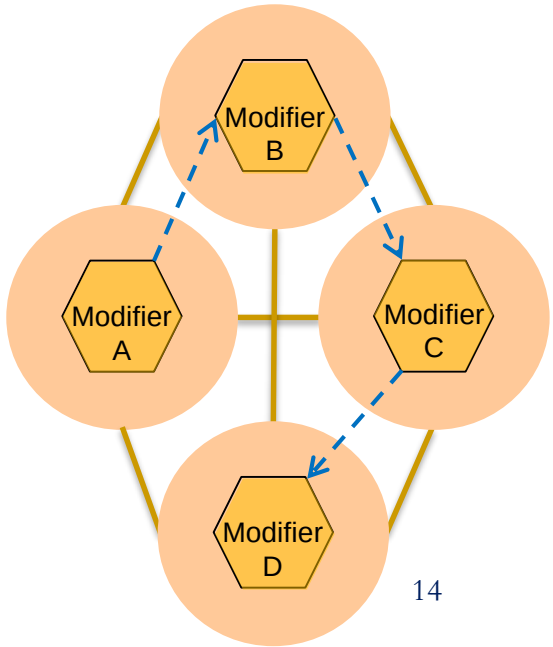
- Déploiement C



SFC prétraitée



MILP de placement et de chaînage



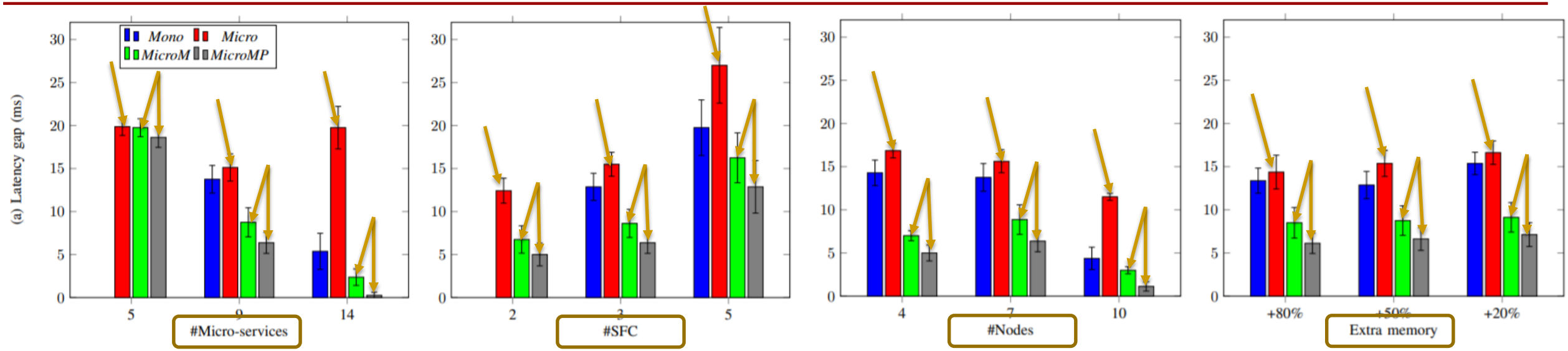
*MILP: Mixed Integer Linear Program

Scénarios d'évaluation

- Approches comparées
  - Placement monolithique de VNF *(Mono)*
  - Placement de micro-services (*Micro*)
  - Placement de micro-services avec mutualisation (*MicroM*)
  - Placement de micro-services avec mutualisation et parallélisation (*MicroMP*) qui correspondent à notre approche
- Paramètres d'évaluation
  - Composition de la demande SFC = 2-5 VNF / 5-14 micro-services [6]
  - Latence SFC = 5-10ms
  - Latence de traitement des VNF / micro-services = 4ms/1ms
  - Latence de la liaison = 1ms
  - Temps de calcul = 600s avec le solveur CPLEX
  - Capacité de mémoire de l'infrastructure = 1,5 x la capacité requise par les demandes SFC

[6] A. Mouaci, É. Gourdin, I. LjubiĆ and N. Perrot, "Virtual Network Functions Placement and Routing Problem: Path formulation," 2020 in IFIP Networking Conference (Networking), 2020, pp. 55-63.

Analyse des résultats : écart de latence



- *Micro* a le pire écart de latence
  - La décomposition "brute" en micro-services augmente le nombre d'entités déployées et donc la communication au sein du réseau.
- *MicroMP* obtient les meilleures performances
  - La mutualisation réduit la longueur des SFC
  - La parallélisation réduit la latence d'exécution
- Le gain de latence est corrélé à la longueur des SFC.
  - Lorsque les SFC sont plus grands, les possibilités de mutualisation et de parallélisation sont plus grandes

Sommaire

- Services à très faible latence et approche micro-services
- Modélisation de l'approche de placement et de chaînage des micro-services
- Vers un placement équilibré des services à très faible latence et de services Best Effort
- Approche de résolution heuristique
- Conclusion et travaux futurs

Cohabitations des SFC Low Latency(LL) et Best Effort(BE)

- Les SFC Best Effort (BE) : demandes de service n'ayant pas de contraintes de latence forte

- La cohabitation avec des SFC LL nécessite une certaine attention
  - Eviter de détériorer la latence des LL

- Porter le principe de gestion des flux de la technologie *L4S* à l'orchestration de micro-services
  - Développement de 3 stratégies de cohabitations

- *L4S*[7] est une technologie qui vise à fournir un haut débit et une faible latence
  - Basée sur un mécanisme ECN permettant de signaler les congestions
  - Réduction du débit pour les deux types de trafic LL et BE en cas de congestion
  - Partage la totalité des ressources disponibles à l'ensemble des paquets

- A l'inverse du «*Slicing*» promulguant une séparation des ressources réseaux pour chaque classe

[7] D. BoruOljira, K.-J. Grinnemo, A. Brunstrom, and J. Taheri, "Validating the sharing behavior and latency characteristics of the l4s architecture," SIGCOMM Comput. Commun. Rev., vol. 50, no. 2, p. 37–44, may 2020. [Online]. Available: https://doi.org/10.1145/3402413.3402419

Stratégies de cohabitation

- Les stratégies proposées sont définies par deux critères : Latence et Déploiement

Stratégie	Paramètres	BE	LL
Fair	Latence	Minimiser	Minimiser
	Déploiement	Partiel	Partiel
LatRes	Latence	Minimiser	Respect stricte
	Déploiement	Partiel	Partiel
LLDep	Latence	Minimiser	Minimiser
	Déploiement	Partiel	Total

- Introduction dans la fonction objectif d'une variable $\Omega \in [0,1]$
- Ω permet une pondération de l'importance attribuée au déploiement de chaque classe de services
- Exemple : fonction objectif de la stratégie *Fair*

$$Min\ ((latence(LL) + nombre\ de\ LL\ non\ déployées)\ *\ \Omega + (latence(BE) + nombre\ de\ BE\ non\ déployées) * (1\ -\ \Omega))$$

Stratégies de cohabitation : évaluation

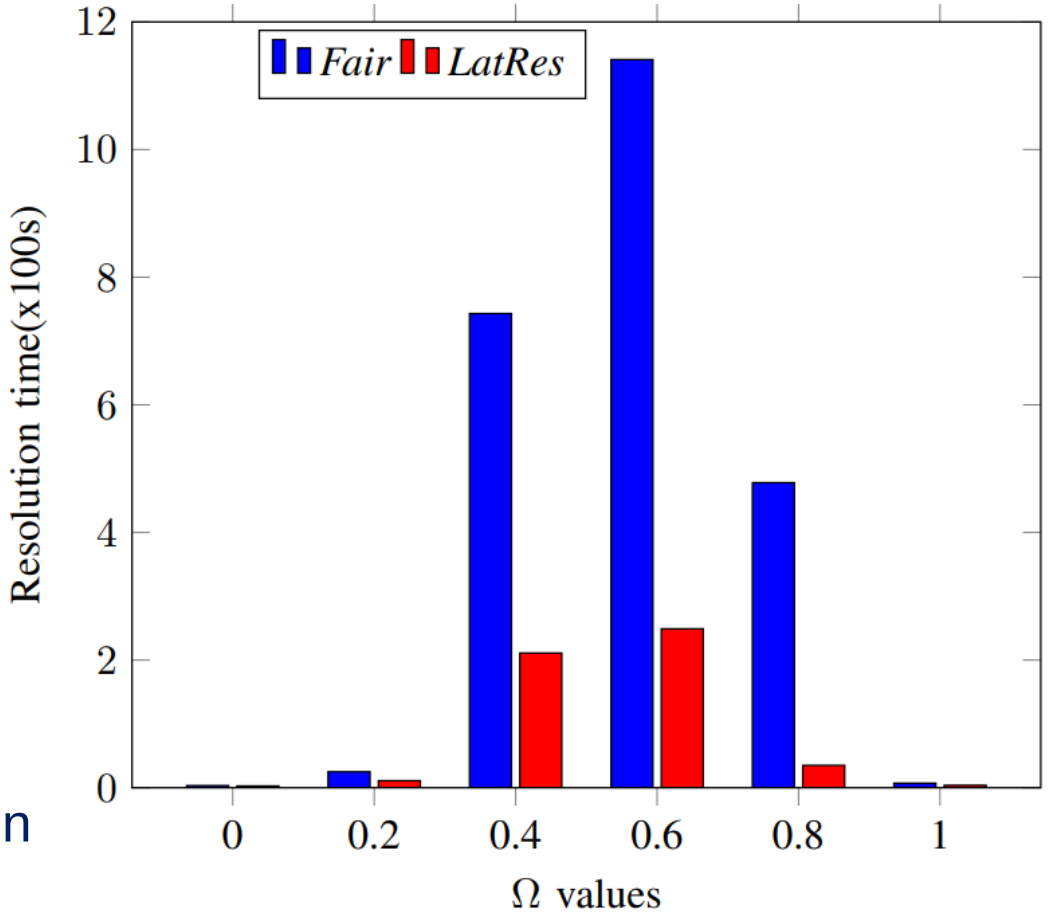
- Test de déploiement
  - #SFC : 4 LL et 3 BE; Latence requise : 10-15ms

	Stratégies *Fair* et *LatRes*				Stratégie *LLDep*			
Ω	#LL	#BE	%Res LL	Lat moy BE	#LL	#BE	%Res LL	Lat moy BE
0	0	3	0%	7,33	4	1	100%	6
0,2	2	3	100%	7,33	4	1	100%	6
0,4	3	2	100%	7	4	1	100%	6
0,6	4	1	100%	6	4	1	100%	6
0,8	4	1	100%	6	4	1	100%	6
1	4	0	100%	-	4	0	100%	-

- Les stratégies Fair et LatRes ont des résultats équivalents
- Le déploiement de BE n'impacte pas la latence des LL
- Le modèle déploie les LL de manière optimale : Ω permet uniquement de faire varier le nombre de SFC déployées

Stratégies de cohabitation : évaluation

- Evaluation du temps de calcul
  - #Noeud : 4 nœuds
  - #SFC : 4 LL et 3 BE;
  - Latence requise : 10-15ms
- Analyse
  - L'approche *Fair* nécessite entre 3 et 6 fois plus de temps que l'approche *LatRes*
  - L'approche *LatRes* oblige le déploiement de l'ensemble des SFC LL ce qui réduit le nombre de solutions possibles
  - Le temps de résolution sur cette configuration peut atteindre 20h pour certaines valeurs de Ω



- Ces temps de résolution ne permettent pas une utilisation réaliste du modèle développé
- Nécessité de réfléchir à une approche de résolution plus rapide…!

Sommaire

- Services à très faible latence et approche micro-services
- Modélisation de l'approche de placement et de chaînage des micro-services
- Vers une placement équilibré des services à très faible latence et de services Best Effort
- Approche de résolution heuristique
- Conclusion et travaux futurs

Etat de l'art : heuristique de placement et de chainage

- Un nombre important d'heuristiques permettant la résolution de la problématique de placement et de chainage traité par la littérature [8]

- [9][10] proposent des heuristiques se basant sur l'algorithme de *Dijkstra*
  - Calcul du plus court chemin
  - Déploiement d'un maximum de VNF jusqu'à saturation du chemin
  - [9] suppression des nœuds saturés et recalcul du plus court chemin
  - [10] calcul d'un chemin alternative en supprimant un des arcs du plus court chemin initial

- La parallélisation n'est pas mise à profit dès la conception de la solution
- Le calcul d'un plus court chemin secondaire n'est pas calculé d'une manière optimale

[8] Jie Sun, Yi Zhang, Feng Liu, Huandong Wang, Xiaojian Xu, Yong Li, "A survey on the placement of virtual network functions", Journal of Network and Computer Applications, Volume 202, 2022, https://doi.org/10.1016/j.jnca.2022.103361.
[9] A. Hirwe and K. Kataoka, "LightChain: A lightweight optimisation of VNF placement for service chaining in NFV," 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 33-37, doi: 10.1109/NETSOFT.2016.7502438.
[10] : Gadre, Akshay & Anbiah, Anix & Sivalingam, Krishna. (2018). Centralized approaches for virtual network function placement in SDN-enabled networks. EURASIP Journal on Wireless Communications and Networking. 2018. 10.1186/s13638-018-1216-0.

Etat de l'art : heuristique de placement et de chainage

- [11] Propose une modélisation de la problématique à travers un ILP
- Développe une heuristique permettant
  - La réduction du temps de calcul de 60 jusqu'à 3500 fois
  - Des solutions entre 10 et 30% de l'optimale

- Modélisation de la problématique sous forme de graphe à plusieurs niveaux
  - Chaque niveau représente le déploiement d'une VNF
  - La valeur des arcs entre les niveaux synthétise un score prenant en compte
    - Le temps de déploiement de la VNF sur le nœud de destination
    - La latence permettant d'atteindre le nœud de destination
- S'inspire de l'algorithme de Viterbi[12] afin de réaliser le déploiement

- La parallélisation n'est pas connu avant le déploiement ce qui rend difficile la création du graphe à plusieurs niveau

[11] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," in IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 725-739, Dec. 2016, doi: 10.1109/TNSM.2016.2569020
[12] G. D. Forney, "The viterbi algorithm," in Proceedings of the IEEE, vol. 61, no. 3, pp. 268-278, March 1973, doi: 10.1109/PROC.1973.9030.

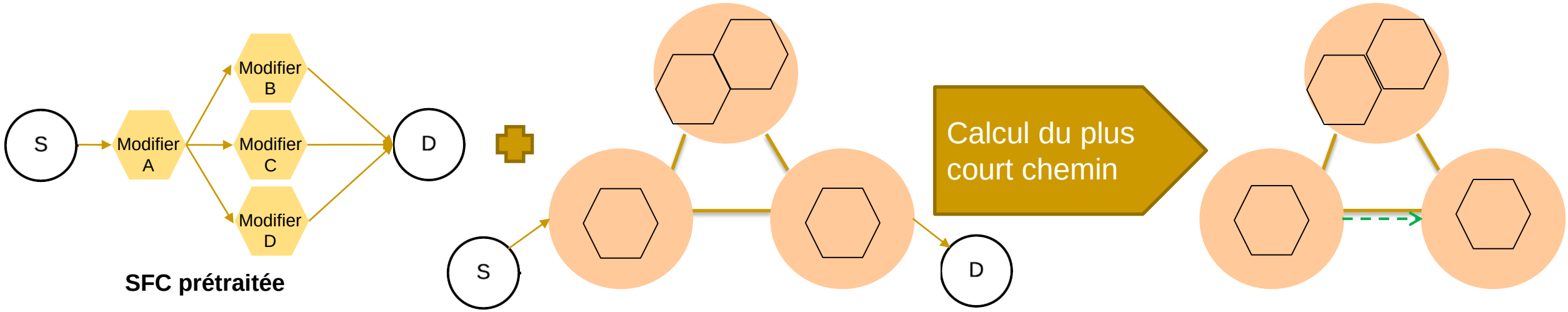
Heuristique de résolution pour les micro-services

- Développement d'une heuristique dédiée au placement et chainage de micro-services
  - Propose des solutions « correctes » en un temps acceptable
  - Basé sur l'approche de calcul des k plus court chemin
  - Gère la parallélisation qui est une forte composante des micro-services

- Notre approche heuristique consiste donc pour chaque SFC :
  - Calculer les k plus courts chemins
  - Tenter de déployer la SFC dessus
  - Si déploiement réussi : vérifier le respect de la latence
    - Si latence non respectée : lancer une phase de parallélisation interne
  - Si déploiement non réussi
    - Relâcher les contraintes de capacité des nœuds
    - Déployer la SFC sur le chemin ayant le plus de capacité
    - Lancer une phase de parallélisation externe afin de soulager les capacités du chemin

Heuristique de résolution pour les micro-services

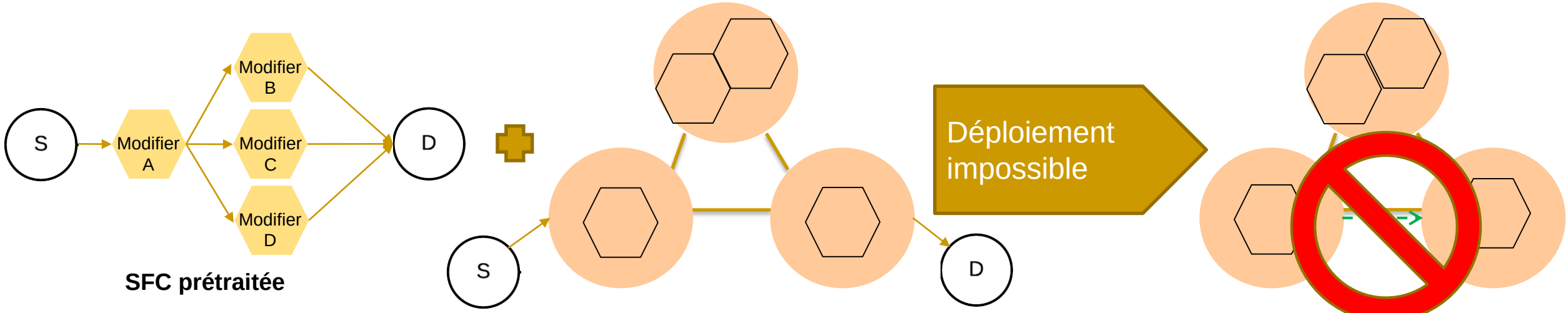
- Comment calculer les k-plus courts chemins ?
  - Utiliser l'algorithme de Dijkstra de manière itérative en supprimant à chaque fois des arcs
    - Méthode non optimale en terme de complexité
  - Utiliser l'algorithme d'Eppstein[13]
  - Permet le calcul des k plus courts chemins avec une complexité *O(m + n log(n) + k)*
    - *m = le nombre d'arc, n = le nombre de noeud*
    - Calcule une représentation des k plus courts chemins à partir de laquelle nous pouvons lire le k^{ème} plus court chemin
- Exemple du fonctionnement de l'algorithme



[13] David Eppstein. 1998. Finding the k Shortest Paths. SIAM J. Comput. 28, 2 (1998), 652–673. https://doi.org/10.1137/S0097539795290477

Heuristique de résolution pour les micro-services

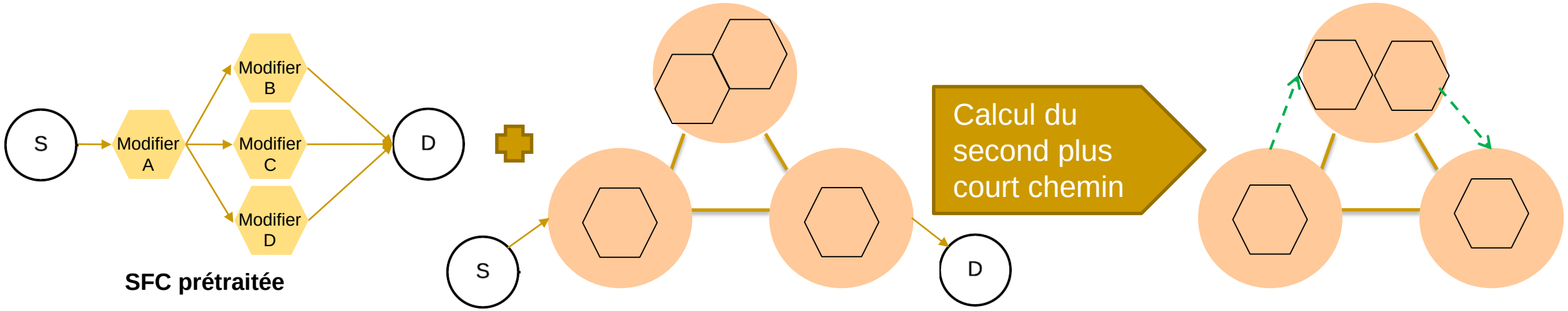
- Comment calculer les k-plus courts chemins ?
  - Utiliser l'algorithme de Dijkstra de manière itérative en supprimant à chaque fois des arcs
    - Méthode non optimale en terme de complexité
  - Utiliser l'algorithme d'Eppstein[13]
  - Permet le calcul des k plus courts chemins avec une complexité *O(m + n log(n) + k)*
    - *m = le nombre d'arc, n = le nombre de noeud*
    - Calcule une représentation des k plus courts chemins à partir de laquelle nous pouvons lire le k^{ème} plus court chemin
- Exemple du fonctionnement de l'algorithme



[13] David Eppstein. 1998. Finding the k Shortest Paths. SIAM J. Comput. 28, 2 (1998), 652–673. https://doi.org/10.1137/S0097539795290477

Heuristique de résolution pour les micro-services

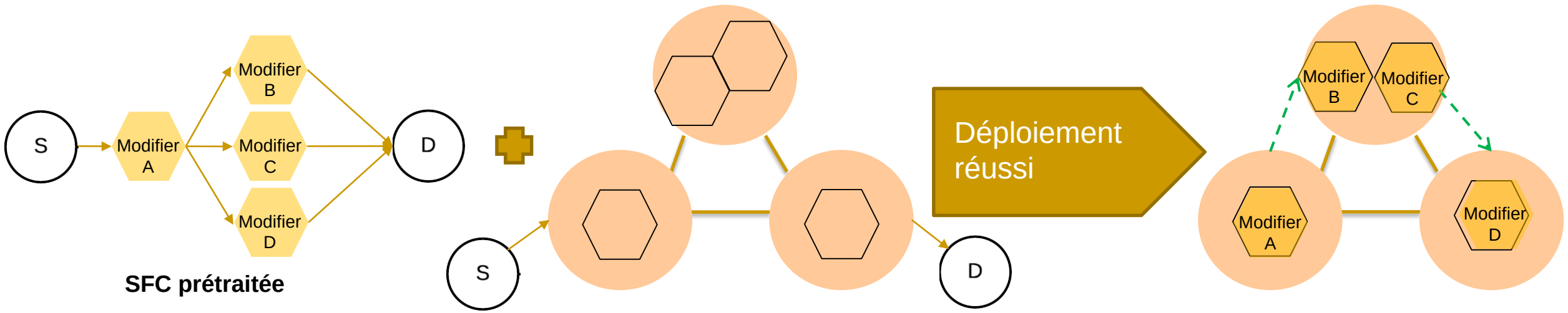
- Comment calculer les k-plus courts chemins ?
  - Utiliser l'algorithme de Dijkstra de manière itérative en supprimant à chaque fois des arcs
    - Méthode non optimale en terme de complexité
  - Utiliser l'algorithme d'Eppstein[13]
  - Permet le calcul des k plus courts chemins avec une complexité *O(m + n log(n) + k)*
    - *m = le nombre d'arc, n = le nombre de noeud*
    - Calcule une représentation des k plus courts chemins à partir de laquelle nous pouvons lire le k^{ème} plus court chemin
- Exemple du fonctionnement de l'algorithme



[13] David Eppstein. 1998. Finding the k Shortest Paths. SIAM J. Comput. 28, 2 (1998), 652–673. https://doi.org/10.1137/S0097539795290477

Heuristique de résolution pour les micro-services

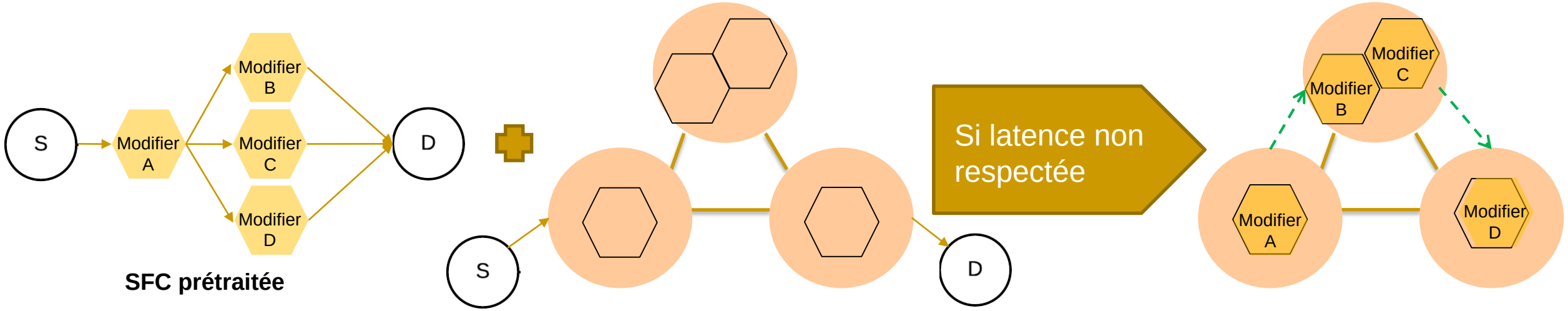
- Comment calculer les k-plus courts chemins ?
  - Utiliser l'algorithme de Dijkstra de manière itérative en supprimant à chaque fois des arcs
    - Méthode non optimale en terme de complexité
  - Utiliser l'algorithme d'Eppstein[13]
  - Permet le calcul des k plus courts chemins avec une complexité *O(m + n log(n) + k)*
    - *m = le nombre d'arc, n = le nombre de noeud*
    - Calcule une représentation des k plus courts chemins à partir de laquelle nous pouvons lire le k^{ème} plus court chemin
- Exemple du fonctionnement de l'algorithme



[13] David Eppstein. 1998. Finding the k Shortest Paths. SIAM J. Comput. 28, 2 (1998), 652–673. https://doi.org/10.1137/S0097539795290477

Heuristique de résolution pour les micro-services

- Comment calculer les k-plus courts chemins ?
  - Utiliser l'algorithme de Dijkstra de manière itérative en supprimant à chaque fois des arcs
    - Méthode non optimale en terme de complexité
  - Utiliser l'algorithme d'Eppstein[13]
  - Permet le calcul des k plus courts chemins avec une complexité *O(m + n log(n) + k)*
    - *m = le nombre d'arc, n = le nombre de noeud*
    - Calcule une représentation des k plus courts chemins à partir de laquelle nous pouvons lire le k^{ème} plus court chemin
- Exemple du fonctionnement de l'algorithme



[13] David Eppstein. 1998. Finding the k Shortest Paths. SIAM J. Comput. 28, 2 (1998), 652–673. https://doi.org/10.1137/S0097539795290477

Sommaire

- Services à très faible latence et approche micro-services
- Modélisation de l'approche de placement et de chaînage des micro-services
- Vers un placement équilibrée des services à très faible latence et de services Best Effort
- Approche de résolution heuristique
- Conclusion et travaux futurs

Conclusion et travaux futurs

- Avantages de l'approche développée
  - Mutualisation des micro-services permettant une réduction de la longueur des SFC
  - Gestion des bifurcations et des fusions dans le chaînage
  - Optimisation du parallélisme en fonction de l'infrastructure, ce qui permet d'obtenir de meilleurs résultats en matière de déploiement
  - Gestion de la cohabitation des SFC LL et BE

- Travaux futurs
  - Terminer l'implémentation de l'heuristique proposée
  - Porter les stratégies de cohabitations à l'heuristique
  - Utiliser l'heuristique afin de développer une méta-heuristique permettant d'atteindre une solution proche de l'optimale dans un temps acceptable
  - Intégrer l'allocation dynamique des ressources au modèle mathématique
    - Si nous allouons plus de ressources à un micro-service, la latence d'exécution sera réduite

Merci pour votre attention

Questions ?

References

- [1] Z. Meng, J. Bi, H. Wang, C. Sun and H. Hu, "MicroNF: An Efficient Framework for Enabling Modularized Service Chains in NFV," in IEEE Journal on Selected Areas in Communications, Aug. 2019
- [2] S. R. Chowdhury, Anthony, H. Bian, T. Bai and R. Boutaba, "A Disaggregated Packet Processing Architecture for Network Function Virtualization" in IEEE Journal on Selected Areas in Communications, vol. 38, no. 6, pp. 1075-1088, June 2020
- [3] Yang Zhang, Bilal Anwer, Vijay Gopalakrishnan, Bo Han, Joshua Reich, Aman Shaikh, and Zhi-Li Zhang. 2017. ParaBox: Exploiting Parallelism for Virtual Network Functions in Service Chaining. In Proceedings of the Symposium on SDN Research (SOSR '17). Association for Computing Machinery, New York, NY, USA.
- [4] Chen Sun, Jun Bi, Zhilong Zheng, Heng Yu, and Hongxin Hu. 2017. NFP: Enabling Network Function Parallelism in NFV. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 43–56.
- [5] H. Magnouche, G. Doyen and C. Prodhon, "Leveraging Micro-Services for Ultra-Low Latency: An optimization Model for Service Function Chains Placement," 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), 2022, pp. 198-206, doi: 10.1109/NetSoft54395.2022.9844040.
- [6] A. Mouaci, É. Gourdin, I. LjubiĆ and N. Perrot, "Virtual Network Functions Placement and Routing Problem: Path formulation," 2020 in IFIP Networking Conference (Networking), 2020, pp. 55-63.
- [7] D. BoruOljira, K.-J. Grinnemo, A. Brunstrom, and J. Taheri, “Validating the sharing behavior and latency characteristics of the l4s architecture,” SIGCOMM Comput. Commun. Rev., vol. 50, no. 2, p. 37–44, may 2020. [Online]. Available: https://doi.org/10.1145/3402413.3402419
- [8] Jie Sun, Yi Zhang, Feng Liu, Huandong Wang, Xiaojian Xu, Yong Li, “A survey on the placement of virtual network functions”, Journal of Network and Computer Applications, Volume 202, 2022, https://doi.org/10.1016/j.jnca.2022.103361.

References

- [9] A. Hirwe and K. Kataoka, "LightChain: A lightweight optimisation of VNF placement for service chaining in NFV," 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 33-37, doi: 10.1109/NETSOFT.2016.7502438.
- [10] : Gadre, Akshay & Anbiah, Anix & Sivalingam, Krishna. (2018). Centralized approaches for virtual network function placement in SDN-enabled networks. EURASIP Journal on Wireless Communications and Networking. 2018. 10.1186/s13638-018-1216-0.
- [11] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," in IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 725-739, Dec. 2016, doi: 10.1109/TNSM.2016.2569020
- [12] G. D. Forney, "The viterbi algorithm," in Proceedings of the IEEE, vol. 61, no. 3, pp. 268-278, March 1973, doi: 10.1109/PROC.1973.9030.
- [13] David Eppstein. 1998. Finding the k Shortest Paths. SIAM J. Comput. 28, 2 (1998), 652–673. https://doi.org/10.1137/S0097539795290477

Annexe A : SFC utilisé dans l'évaluation

- SFC composées de
  - Firewall : Read, Header Classifier, Alert, Drop, Output
  - NAT : Read, Header Classifier, Modifier, Output
  - Traffic monitor : Read, CheckIP Header, http Classifier, Output
  - IPS : Read, Payloead Classifier, Header Classifier, Count URL

- Chaque SFC est composée de 2 à 5 VNF.

Annexe B : diagramme heuristique de déploiement