Markov Approximation Method for Optimal Service Orchestration in IoT Network

WENCHEN HE, SHAOYONG GUO1, YUN LIANG, AND XUESONG QIU1

Présenté par Baptiste Viera

ÉTS
Le génie pour l'industrie

INTRODUCTION

ÉTS
Le génie pour l'industrie

CONTEXTE

- **Evolution : IoT & 5G**:
  - L'edge computing, les NFV, les méthodes distribuées, et les exigences en matière de qualité d'expérience (QoE) sont et seront essentiels à l'avenir.
  - Il existe un réel besoin de proposer des solutions efficientes en terme de scalabilité, flexibilité et de ressources
- **Travaux sur le placement de VNF dans l'IoT :**
  - Détermination du nombre optimal de VNF, du placement de VNF avec plusieurs instances, du chemin d'acheminement optimal du trafic avec des méthodes centralisées

ÉTS
Le génie pour l'industrie

PROBLEMATIQUE

- Le placement de VNFs avec plusieurs instances est un problème NP-difficile
- Les algorithmes centralisés ont besoin d'énormes ressources de calcul et de mémoire pour une seule machine physique
- Les algorithmes de placement aléatoire et à chemin unique sont couteux

OBJECTIFS

- Présenter une méthode distribuée pour le placement optimal des VNFs avec plusieurs instances afin de :
  - Minimiser les coûts et les délais
  - Garantir l'équilibrage de la charge du réseau

METHODOLOGIE

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

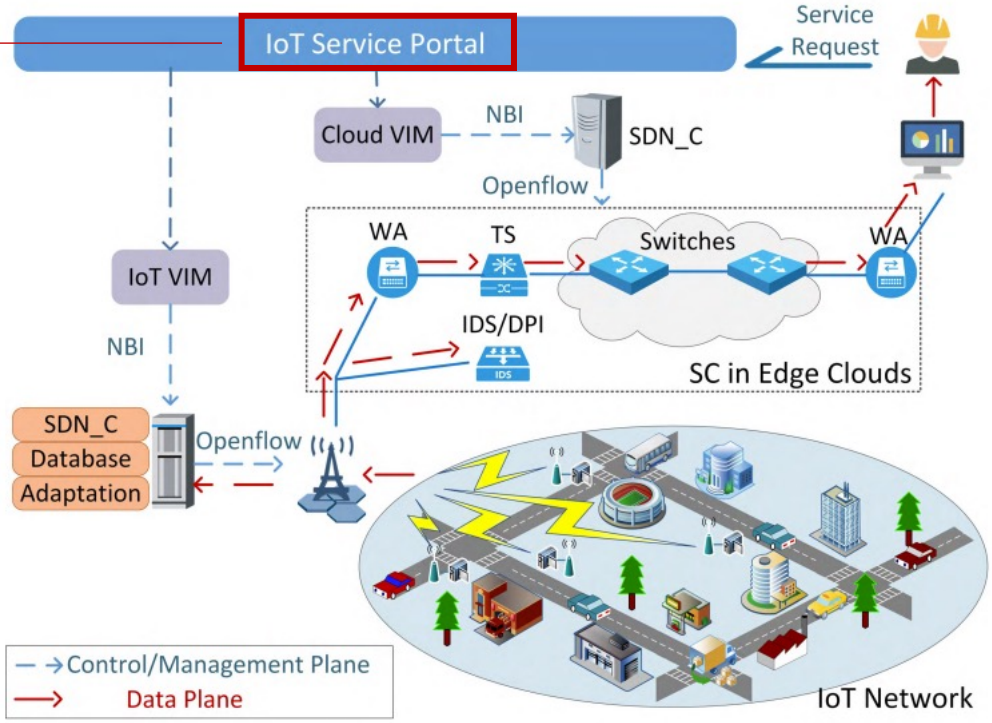


FIGURE 1. IoT service provisioning architecture.

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

Orchestrer divers modules de fonctions réseau

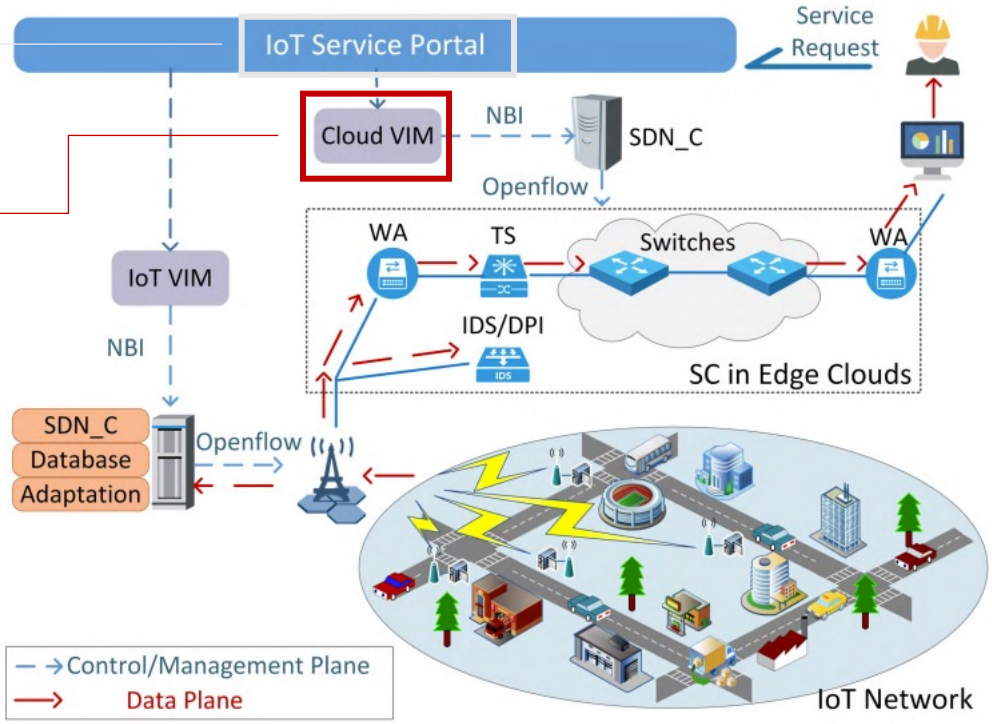


FIGURE 1. IoT service provisioning architecture.

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

Orchestrer divers modules de fonctions réseau

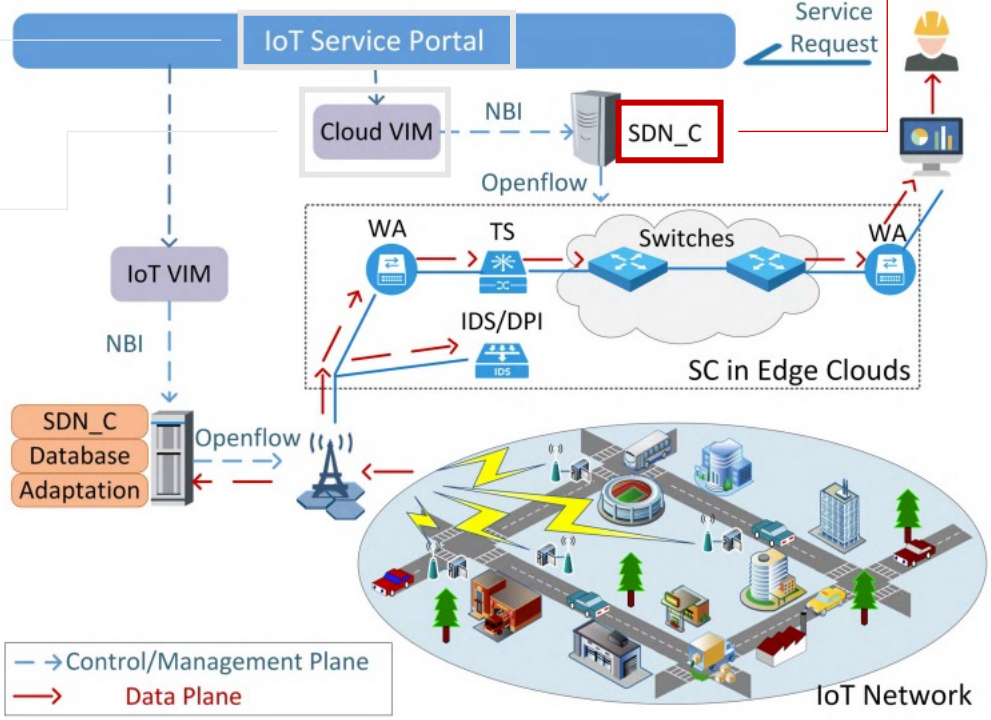


FIGURE 1. IoT service provisioning architecture.

Placer et gérer les VNF et ordonnancer le trafic entre les serveurs de l'edge

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

Orchestrer divers modules de fonctions réseau

Gérer les composants et les ressources du réseau IoT

Placer et gérer les VNF et ordonnancer le trafic entre les serveurs de l'edge

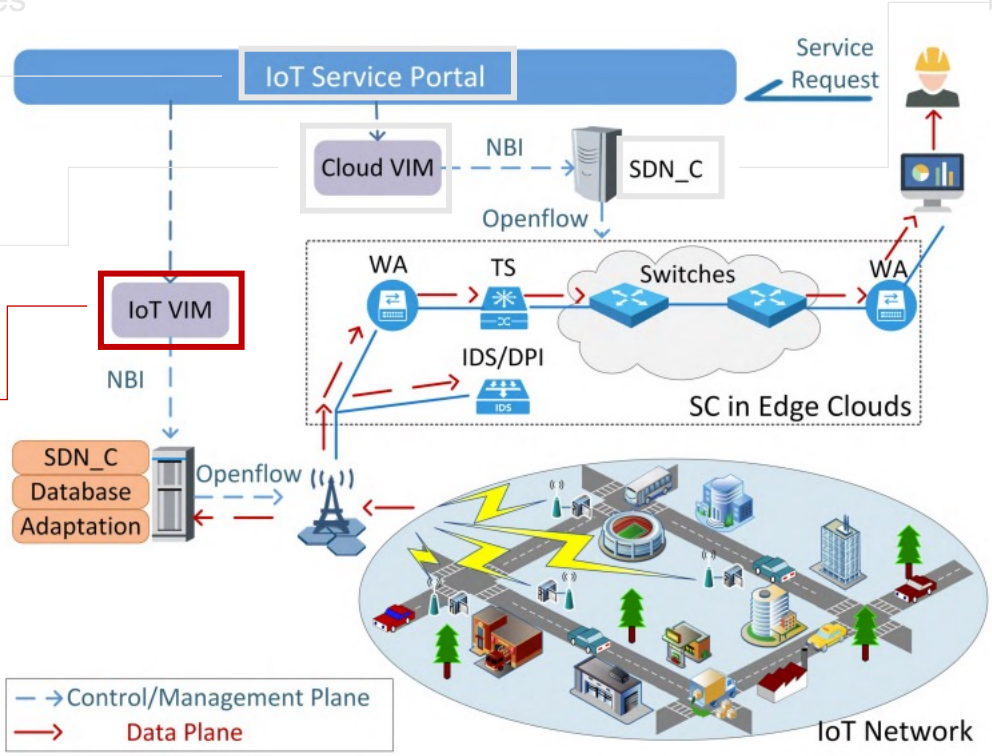


FIGURE 1. IoT service provisioning architecture.

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

Orchestrer divers modules de fonctions réseau

Gérer les composants et les ressources du réseau IoT

Placer et gérer les VNF et ordonnancer le trafic entre les serveurs de l'edge

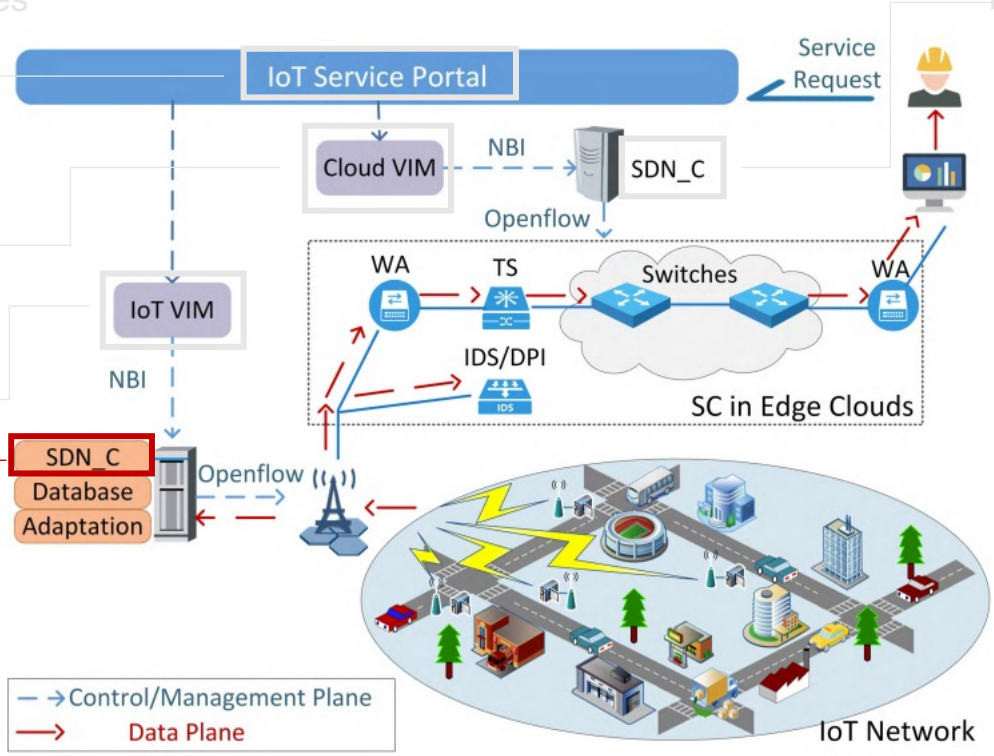


FIGURE 1. **IoT service provisioning architecture.**

Exécution du plan de contrôle « défini par logiciel » de la structure IoT

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

Orchestrer divers modules de fonctions réseau

Gérer les composants et les ressources du réseau IoT

Exécution du plan de contrôle « défini par logiciel » de la structure IoT

Placer et gérer les VNF et ordonnancer le trafic entre les serveurs de l'edge

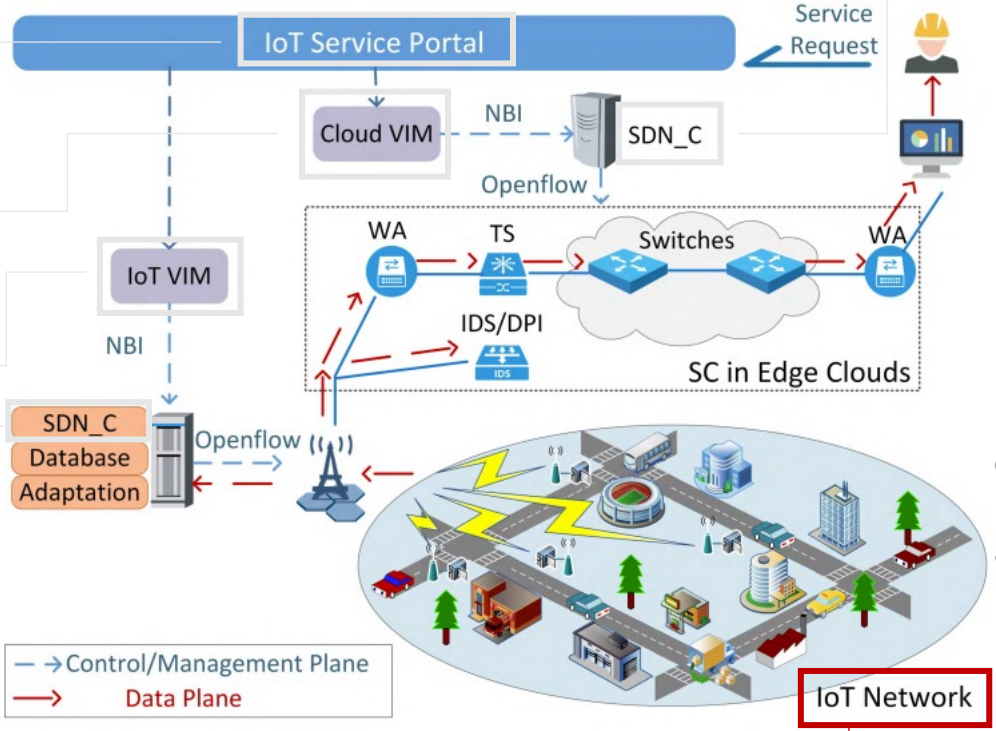


FIGURE 1. IoT service provisioning architecture.

Ressources d'infrastructure du Radio Access Network (RAN), des capteurs et des actionneurs

IoT Network

Prendre en charge la collecte des données

ARCHITECTURE DU SYSTÈME ET DU MODELE

Recevoir les requêtes et fournir divers services aux utilisateurs de l'IoT

Orchestrer divers modules de fonctions réseau

Gérer les composants et les ressources du réseau IoT

Exécution du plan de contrôle « défini par logiciel » de la structure IoT

Placer et gérer les VNF et ordonnancer le trafic entre les serveurs de l'edge

Traiter les données par des VNF ordonnés et les envoyer aux utilisateurs

Ressources d'infrastructure du Radio Access Network (RAN), des capteurs et des actionneurs

Prendre en charge la collecte des données

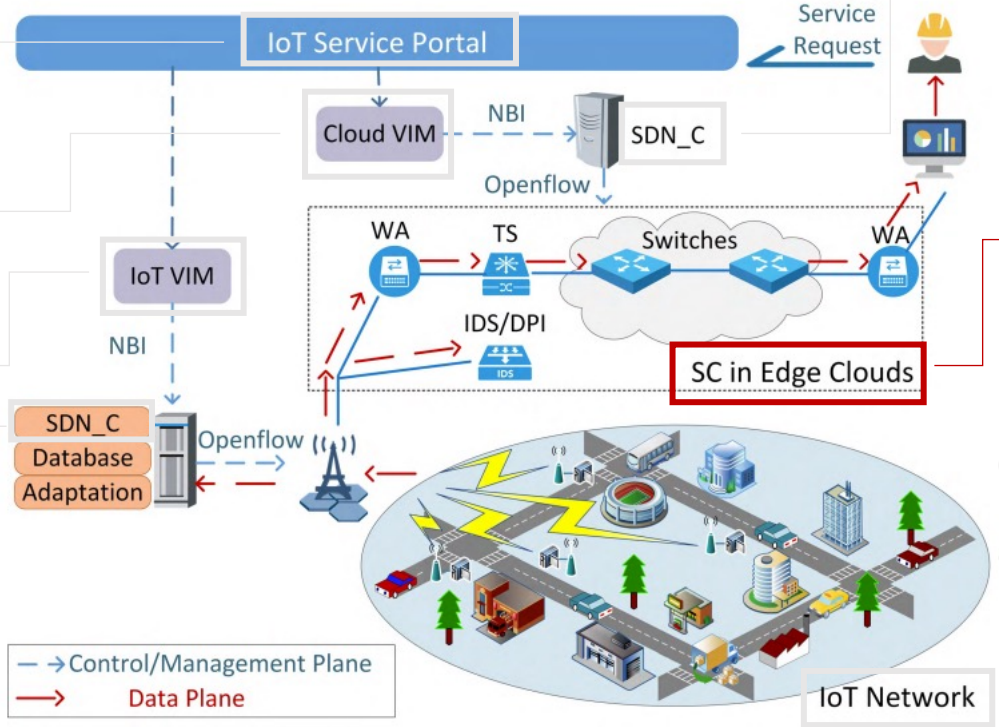


FIGURE 1. IoT service provisioning architecture.

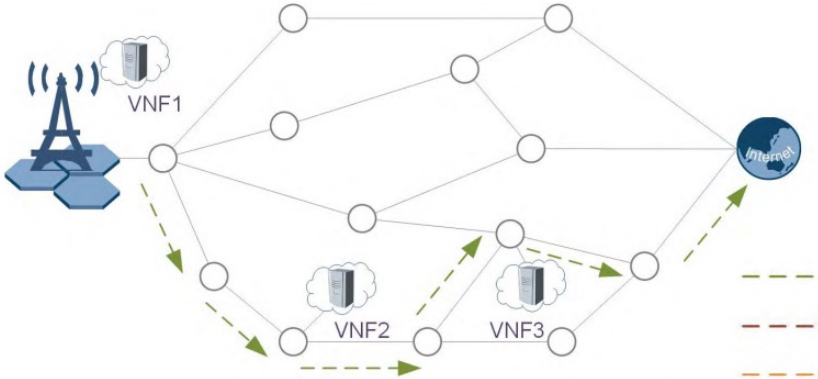
ARCHITECTURE DU SYSTÈME ET DU MODELE

- **v** : les serveurs pour placer les VNFs avec une certaine capacité CPU et de mémoire
- **lij** : les liens physiques de connections avec un taux de transmission de données maximal (**bij**) et un délai de transmission (**dij**)
- **s** : regroupement de la source et la destination avec les informations de la SC et le taux de transmission

TABLE 1. Definition of variables.

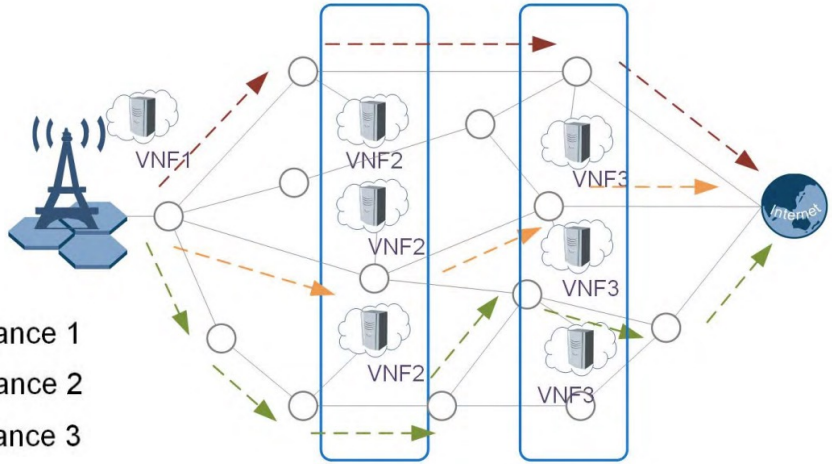
Symbol	Description
$G = (N, L)$	weighted undirected graph
$v \in V$	cloud server nodes
$f \in F_s$	VNF f in SC s
r_s	data transmission rate of SC request s
$Cap_{cpu}(v)$	CPU resource in server v
$Cap_{mem}(v)$	memory resource in server v
$Cap(v)_{remain}$	weighted sum of remaining resources in server v
cpu_f	required CPU by VNF f
mem_f	required memory resource by VNF f
d_f	processing delay of VNF f
l_{ij}	physical link
l^v_{uw}	virtual link between VNFs u and w
b_{ij}	data transmission rate of l_{ij}
b^{ij}_{remain}	remaining transmission rate of l_{ij}
d_{ij}	transmission delay of l_{ij}
Φ_v	load balancing factor of server v
Θ_{ij}	load balancing factor of link l_{ij}
$\alpha_1, \beta_1, \lambda_1$	adjustment parameters for Φ_v
$\alpha_2, \beta_2, \lambda_2$	adjustment parameters for Θ_{ij}
D_s	minimum delay requirement
c_1, c_2, c_3	unit price of CPU, memory and transmission rate
$Cost_{total}$	total network cost
$(v_{so}, v_{de})_s$	source and destination node pair of s
z_p, z_m, z_d, z_l	weights of CPU, memory, delay and load balancing factor
Binary Variable	**Description**
$x^s_{v,f}$=1	VNF f in s is mapped to server v
$x^s_{v,f'}$=1	VNF replica f' of f in s is mapped to server v
$y^s_{ij,uw}$=1	l^v_{uw} in s is mapped to physical link l_{ij}

ILLUSTRATION DU PROBLEME D'INSTANCE UNIQUE



VS

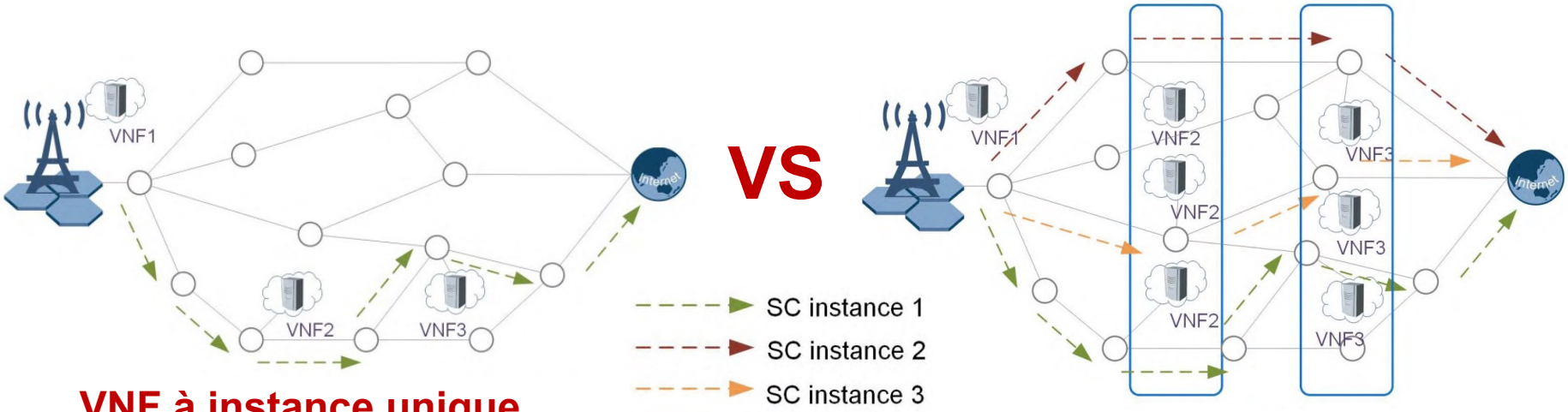
SC instance 1
SC instance 2
SC instance 3



VNF à instance unique

VNF à multiples uniques

ILLUSTRATION DU PROBLEME D'INSTANCE UNIQUE



VNF à instance unique

VNF à multiple unique

TABLE 2. Performance of two schemes.

scheme	forwarding cost	placement cost	delay	variance
(a)	120Mbps −	3 +	60ms −	0.1 −
(b)	60Mbps +	7 −	40ms +	0.006 +
Ratio (a//b)	/ 2	* 2.3	/ 1.5	/ 16.7

PLACEMENT VNF AVEC PLUSIEURS INSTANCES.

CONTRAINTES DE CAPACITÉ

- Héberger des instances VNF contraint par les ressources CPU et de mémoire.
- Transmission du trafic de service contraint par le débit de transmission de données.
- Placement sur le même serveur d'une instance VNF f et sa réplique f impossible.
- Remplacement par une seule de 2 instances VNF identiques sur le même serveur
- Réplication indéfinie d'une instance VNF impossible
- Contrainte de conservation du flux

ÉTS
Le génie pour l'industrie

PLACEMENT VNF AVEC PLUSIEURS INSTANCES.

CONTRAINTE DE RETARD

$$C_7 : \sum_{v \in V} \sum_{f \in F_s} x^s_{v,f} \cdot d_f + \sum_{i,j \in N} \sum_{u,w \in F_s} y^s_{ij,uw} \cdot d_{ij} \leq D_s$$

Délai de traitement sur les VNF

Délai de transmission sur les liaisons

Délai minimum

PLACEMENT VNF AVEC PLUSIEURS INSTANCES.

MODÈLE D'OPTIMISATION MULTI-OBJECTIF

$$\boldsymbol{Min}\ \{cost(server) + cost(link)\}$$

En prenant en considérations le contraintes précédentes

- Améliorer l'efficacité des coûts
- Garantir l'équilibrage des charges

ÉTS
Le génie pour l'industrie

DESCRIPTION DE L'ALGORITHME

INTRODUCTION

- Problème de mappage SC :
  - Un problème d'optimisation combinatoire NP-difficile

- Solution:
  - Approximer le modèle d'optimisation pour le résoudre de façon distribuée
  - Proposer un algorithme de placement de VNF à instances multiples basé sur la chaîne de Markov.

DESCRIPTION DE L'ALGORITHME

PLACEMENT DE VNF AVEC PLUSIEURS INSTANCES GRACE A UN ALGORITHME BASÉ SUR LA CHAÎNE DE MARKOV

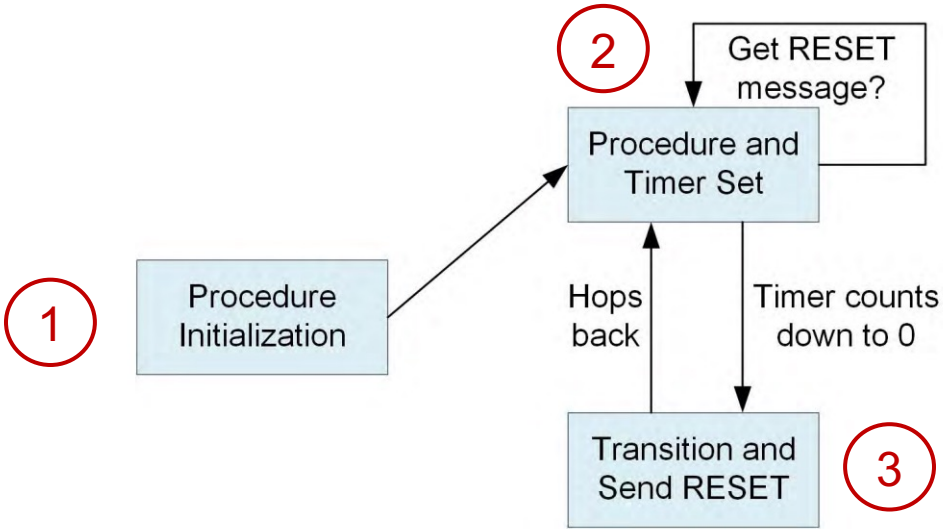


FIGURE 3. General state machine.

DESCRIPTION DE L'ALGORITHME

INITIALISATION DE LA PROCEDURE

Algorithm 1 Procedure Initialization

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: Initialization scheme;

1: Weights of links are initialized as $c_3 \cdot (z_l \cdot \Theta_{ij} + z_d \cdot d_{ij})$
2: **for** $s \in S$ **do**;
3: Calculate the feasible path between $(v_{so}, v_{de})_s$ by shortest path algorithm;
4: Calculate available servers in current path;
5: **if** the number of available servers is not enough **then**
6: Search for available servers closest to current path;
7: Extend path to new servers;
8: **end if**
9: Place randomly VNF instances $f \in s$ in the available servers;
10: Get the action set $\{x^s_{f,v}\}$ and $\{y^s_{ij,uw}|x\}$ of s;
11: Calculate $Cost(a)$ based on (6);
12: **end for**
13: **return** $\{x^s_{f,v}\}$, $\{y^s_{ij,uw}|x\}$, $Cost(a)$;

→ Allocation des threads dédiés pour chaque SC

DESCRIPTION DE L'ALGORITHME

INITIALISATION DE LA PROCEDURE

Algorithm 1 Procedure Initialization

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: Initialization scheme;

1: Weights of links are initialized as $c_3 \cdot (z_l \cdot \Theta_{ij} + z_d \cdot d_{ij})$

2: **for** $s \in S$ **do**;

3: Calculate the feasible path between $(v_{so}, v_{de})_s$ by shortest path algorithm;

4: Calculate available servers in current path;

5: **if** the number of available servers is not enough **then**

6: Search for available servers closest to current path;

7: Extend path to new servers;

8: **end if**

9: Place randomly VNF instances $f \in s$ in the available servers;

10: Get the action set $\{x^s_{f,v}\}$ and $\{y^s_{ij,uw}|x\}$ of s;

11: Calculate $Cost(a)$ based on (6);

12: **end for**

13: **return** $\{x^s_{f,v}\}$, $\{y^s_{ij,uw}|x\}$, $Cost(a)$;

Allocation des threads dédiés pour chaque SC

Poids des liens définis par la somme pondérée du délai et du facteur d'équilibrage de la charge

ÉTS Le génie pour l'industrie

DESCRIPTION DE L'ALGORITHME

INITIALISATION DE LA PROCEDURE

Algorithm 1 Procedure Initialization

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: Initialization scheme;

1: Weights of links are initialized as $c_3 \cdot (z_l \cdot \Theta_{ij} + z_d \cdot d_{ij})$

2: **for** $s \in S$ **do**;

3: Calculate the feasible path between $(v_{so}, v_{de})_s$ by shortest path algorithm;

4: Calculate available servers in current path;

5: **if** the number of available servers is not enough **then**

6: Search for available servers closest to current path;

7: Extend path to new servers;

8: **end if**

9: Place randomly VNF instances $f \in s$ in the available servers;

10: Get the action set $\{x^s_{f,v}\}$ and $\{y^s_{ij,uw}|x\}$ of s;

11: Calculate $Cost(a)$ based on (6);

12: **end for**

13: **return** $\{x^s_{f,v}\}$, $\{y^s_{ij,uw}|x\}$, $Cost(a)$;

Allocation des threads dédiés pour chaque SC

Poids des liens définis par la somme pondérée du délai et du facteur d'équilibrage de la charge

Calcul du chemin le plus court en fonction de la paire de nœuds source et destination

DESCRIPTION DE L'ALGORITHME

INITIALISATION DE LA PROCEDURE

Algorithm 1 Procedure Initialization

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: Initialization scheme;

1: Weights of links are initialized as $c_3 \cdot (z_l \cdot \Theta_{ij} + z_d \cdot d_{ij})$

2: **for** $s \in S$ **do**;

3: Calculate the feasible path between $(v_{so}, v_{de})_s$ by shortest path algorithm;

4: Calculate available servers in current path;

5: **if** the number of available servers is not enough **then**

6: Search for available servers closest to current path;

7: Extend path to new servers;

8: **end if**

9: Place randomly VNF instances $f \in s$ in the available servers;

10: Get the action set $\{x^s_{f,v}\}$ and $\{y^s_{ij,uw}|x\}$ of s;

11: Calculate $Cost(a)$ based on (6);

12: **end for**

13: **return** $\{x^s_{f,v}\}$, $\{y^s_{ij,uw}|x\}$, $Cost(a)$;

Allocation des threads dédiés pour chaque SC

Poids des liens définis par la somme pondérée du délai et du facteur d'équilibrage de la charge

Calcul du chemin le plus court en fonction de la paire de nœuds source et destination

Extension du chemin actuel aux serveurs les plus proches jusqu'à ce que le placement de toutes les instances de VNF soit terminé.

DESCRIPTION DE L'ALGORITHME

INITIALISATION DE LA PROCEDURE

Algorithm 1 Procedure Initialization

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: Initialization scheme;

1: Weights of links are initialized as $c_3 \cdot (z_l \cdot \Theta_{ij} + z_d \cdot d_{ij})$

2: **for** $s \in S$ **do**;

3: Calculate the feasible path between $(v_{so}, v_{de})_s$ by shortest path algorithm;

4: Calculate available servers in current path;

5: **if** the number of available servers is not enough **then**

6: Search for available servers closest to current path;

7: Extend path to new servers;

8: **end if**

9: Place randomly VNF instances $f \in s$ in the available servers;

10: Get the action set $\{x^s_{f,v}\}$ and $\{y^s_{ij,uw}|x\}$ of s;

11: Calculate $Cost(a)$ based on (6);

12: **end for**

13: **return** $\{x^s_{f,v}\}$, $\{y^s_{ij,uw}|x\}$, $Cost(a)$;

Allocation des threads dédiés pour chaque SC

Poids des liens définis par la somme pondérée du délai et du facteur d'équilibrage de la charge

Calcul du chemin le plus court en fonction de la paire de nœuds source et destination

Extension du chemin actuel aux serveurs les plus proches jusqu'à ce que le placement de toutes les instances de VNF soit terminé.

Placement aléatoire des instances de VNF dont a besoin « s » dans les serveurs disponibles sur le chemin

DESCRIPTION DE L'ALGORITHME

INITIALISATION DE LA PROCEDURE

Algorithm 1 Procedure Initialization

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: Initialization scheme;

1: Weights of links are initialized as $c_3 \cdot (z_l \cdot \Theta_{ij} + z_d \cdot d_{ij})$

2: **for** $s \in S$ **do**;

3: Calculate the feasible path between $(v_{so}, v_{de})_s$ by shortest path algorithm;

4: Calculate available servers in current path;

5: **if** the number of available servers is not enough **then**

6: Search for available servers closest to current path;

7: Extend path to new servers;

8: **end if**

9: Place randomly VNF instances $f \in s$ in the available servers;

10: Get the action set $\{x^s_{f,v}\}$ and $\{y^s_{ij,uw}|x\}$ of s;

11: Calculate $Cost(a)$ based on (6);

12: **end for**

13: **return** $\{x^s_{f,v}\}$, $\{y^s_{ij,uw}|x\}$, $Cost(a)$;

Allocation des threads dédiés pour chaque SC

Poids des liens définis par la somme pondérée du délai et du facteur d'équilibrage de la charge

Calcul du chemin le plus court en fonction de la paire de nœuds source et destination

Extension du chemin actuel aux serveurs les plus proches jusqu'à ce que le placement de toutes les instances de VNF soit terminé.

Placement aléatoire des instances de VNF dont a besoin « s » dans les serveurs disponibles sur le chemin

Retour du coût de l'étape « a » courant calculé et basé sur la recherche de serveurs, l'ensembles des VNFs et des liaisons

DESCRIPTION DE L'ALGORITHME

PROCEDURE ET REGLAGES DU « TIMER »

Algorithm 2 Procedure and Timer Set

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$

1: **for** $s \in S$ **do**;
2: Choose a VNF f randomly in s;
3: Calculate $\sigma_{f,s}$ for s;
4: **if** $|\sigma_{f,s}| \geq 1$ **then**
5: Choose a replication scheme $x^s_{v,f'} = 1$ form $\sigma_{f,s}$ randomly;
6: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f+1$ and $f-1$;
7: Calculate the new $Cost(a')$ based on $x^s_{v,f'}$ and $(y|x')$;
8: Calculate the timer T_s according to (29);
9: **end if**
10: **end for**
11: **return** $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$;

Sélection aléatoire d'une instance de VNF f à répliquer.

DESCRIPTION DE L'ALGORITHME

PROCEDURE ET REGLAGES DU « TIMER »

Algorithm 2 Procedure and Timer Set

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$

1: **for** $s \in S$ **do**;
2: Choose a VNF f randomly in s;
3: Calculate $\sigma_{f,s}$ for s;
4: **if** $|\sigma_{f,s}| \geq 1$ **then**
5: Choose a replication scheme $x^s_{v,f'} = 1$ form $\sigma_{f,s}$ randomly;
6: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f+1$ and $f-1$;
7: Calculate the new $Cost(a')$ based on $x^s_{v,f'}$ and $(y|x')$;
8: Calculate the timer T_s according to (29);
9: **end if**
10: **end for**
11: **return** $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$;

Sélection aléatoire d'une instance de VNF f à répliquer.

Ensembles des actions de réplication optionnelles pour f .

DESCRIPTION DE L'ALGORITHME

PROCEDURE ET REGLAGES DU « TIMER »

Algorithm 2 Procedure and Timer Set

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$

1: **for** $s \in S$ **do**;
2: Choose a VNF f randomly in s;
3: Calculate $\sigma_{f,s}$ for s;
4: **if** $|\sigma_{f,s}| \geq 1$ **then**
5: Choose a replication scheme $x^s_{v,f'} = 1$ form $\sigma_{f,s}$ randomly;
6: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f+1$ and $f-1$;
7: Calculate the new $Cost(a')$ based on $x^s_{v,f'}$ and $(y|x')$;
8: Calculate the timer T_s according to (29);
9: **end if**
10: **end for**
11: **return** $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$;

Sélection aléatoire d'une instance de VNF f à répliquer.

Ensembles des actions de réplication optionnelles pour f .

Sélection d'un schéma de réplication x aléatoire

DESCRIPTION DE L'ALGORITHME

PROCEDURE ET REGLAGES DU « TIMER »

Algorithm 2 Procedure and Timer Set

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$

1: **for** $s \in S$ **do**;
2: Choose a VNF f randomly in s;
3: Calculate $\sigma_{f,s}$ for s;
4: **if** $|\sigma_{f,s}| \geq 1$ **then**
5: Choose a replication scheme $x^s_{v,f'} = 1$ form $\sigma_{f,s}$ randomly;
6: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f + 1$ and $f - 1$;
7: Calculate the new $Cost(a')$ based on $x^s_{v,f'}$ and $(y|x')$;
8: Calculate the timer T_s according to (29);
9: **end if**
10: **end for**
11: **return** $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$;

Sélection aléatoire d'une instance de VNF f à répliquer.

Ensembles des actions de réplication optionnelles pour f .

Sélection d'un schéma de réplication x aléatoire

Calcul du plus court chemin (y|x') à partir de f' vers ses instances adjacentes f +1 et f -1 et calcul du cout(a')

DESCRIPTION DE L'ALGORITHME

PROCEDURE ET REGLAGES DU « TIMER »

Algorithm 2 Procedure and Timer Set

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;

Output: $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$

1: **for** $s \in S$ **do**;
2: Choose a VNF f randomly in s;
3: Calculate $\sigma_{f,s}$ for s;
4: **if** $|\sigma_{f,s}| \geq 1$ **then**
5: Choose a replication scheme $x^s_{v,f'} = 1$ form $\sigma_{f,s}$ randomly;
6: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f + 1$ and $f - 1$;
7: Calculate the new $Cost(a')$ based on $x^s_{v,f'}$ and $(y|x')$;
8: Calculate the timer T_s according to (29);
9: **end if**
10: **end for**
11: **return** $\sigma_{f,s}$; $\{x^s_{v,f'}\}$; $\{y|x'\}$;

Sélection aléatoire d'une instance de VNF f à répliquer.

Ensembles des actions de réplication optionnelles pour f .

Sélection d'un schéma de réplication x aléatoire

Calcul du plus court chemin (y|x') à partir de f' vers ses instances adjacentes f +1 et f -1 et calcul du cout(a')

Génération d'un « timer » aléatoire exponentiellement distribué Ts pour déterminer s'il faut mettre à jour l'état a' à partir de a.

DESCRIPTION DE L'ALGORITHME

Algorithm 3 VPMIA

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;
Output: $\{x^s_{v,f}\}$; $\{y^s_{ij,uw}\}$; $\{y|x'\}$;

1: Execute **Alg.1**;
2: Execute **Alg.2**;
3: **while** system is running **do**
4: **/*Procedure Transition*/**
5: **for** $s \in S$ **do**;
6: Choose a VNF f randomly in s;
7: Obtain its current placement scheme: $\{x^s_{v,f}\}$ and $\{y^s_{ij,uw}\}$;
8: Choose a new scheme $x^s_{v,f'}$ in $\sigma_{f,s}$;
9: Calculate $Cost(a')$ and T_s;
10: **if** T_s expires **then**
11: Choose $x^s_{v,f'}$ as a replication scheme of f;
12: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f+1$ and $f-1$;
13: Execute **Alg.2;**
14: Spread RESET message to other SC to notify state update;
15: **end if**
16: **/*Procedure RESET*/**
17: **if** SC s receives the RESET message **then**
18: Put $x^s_{v,f'}$ as replication scheme f;
19: Replace $Cost(a)$ with $Cost(a')$;
20: Clear and generate a new timer T_s;
21: **end if**
22: **end for**
23: **end while**
24: **return** $\{x^s_{v,f}\}$; $\{y^s_{ij,uw}\}$; $\{y|x'\}$;

Si le ***timer (Ts)*** expire alors les instances des VNFs et les chemins correspondants sont choisit comme schéma de réplication

Ensuite, ***Procedure et Timer Set (Alg2)*** seront exécutés de manière répétée,

Reset sera envoyé pour notifier la mise à jour de l'état.

DESCRIPTION DE L'ALGORITHME

Algorithm 3 VPMIA

Input: $G = (N, L)$; b_{ij}; d_{ij}; $Cap_{cpu}(v)$; $Cap_{mem}(v)$; $s = \{(v_{so}, v_{de})_s, F_s, r_s\}$;
Output: $\{x^s_{v,f}\}$; $\{y^s_{ij,uw}\}$; $\{y|x'\}$;

1: Execute **Alg.1**;
2: Execute **Alg.2**;
3: **while** system is running **do**
4: **/*Procedure Transition*/**
5: **for** $s \in S$ **do**;
6: Choose a VNF f randomly in s;
7: Obtain its current placement scheme: $\{x^s_{v,f}\}$ and $\{y^s_{ij,uw}\}$;
8: Choose a new scheme $x^s_{v,f'}$ in $\sigma_{f,s}$;
9: Calculate $Cost(a')$ and T_s;
10: **if** T_s expires **then**
11: Choose $x^s_{v,f'}$ as a replication scheme of f;
12: Compute the shortest path $(y|x')$ form f' to its adjacent instances $f+1$ and $f-1$;
13: Execute **Alg.2;**
14: Spread RESET message to other SC to notify state update;
15: **end if**
16: **/*Procedure RESET*/**
17: **if** SC s receives the RESET message **then**
18: Put $x^s_{v,f'}$ as replication scheme f;
19: Replace $Cost(a)$ with $Cost(a')$;
20: Clear and generate a new timer T_s;
21: **end if**
22: **end for**
23: **end while**
24: **return** $\{x^s_{v,f}\}$; $\{y^s_{ij,uw}\}$; $\{y|x'\}$;

Si le ***timer (Ts)*** expire alors les instances des VNFs et les chemins correspondants sont choisit comme schéma de réplication

Ensuite, ***Procedure et Timer Set (Alg2)*** seront exécutés de manière répétée,

Reset sera envoyé pour notifier la mise à jour de l'état.

Mise à jour du timer Ts en fonction du nouveau coût(a) qui sera égale au coût(a')

EVALUATION

MATERIEL

- Simulation d'une plateforme réseau en Java
- Caractéristiques mentionnées ci- dessous

The number of servers	10
The number of forwarding nodes	20
The number of links	50
The CPU of one server	32
The memory resource of server	100-200GB
The transmission rate of link	1Gbps
The required VNFs of SC	2-4
The CPU required by one VNF	2-4
The memory required by one VNF	5-10GB
The data transmission rate of one SC	20-50Mbps
$\alpha_1, \beta_1, \lambda_1$	0.1, 0.1, 0.1
$\alpha_2, \beta_2, \lambda_2$	0.1, 0.1, 0.1
z_p, z_m, z_d, z_l	0.5

TABLE 3. Specific experimental parameters.

ALGORITHMES

- Random Placement Algorithm (RPA)
- Single Path Algorithm (SPA)
- VPMIA

MESURES

- Coût du réseau
- Taux Moyen d'Utilisation des Liens (TMUL)
- Taux Moyen d'Utilisation des Serveurs (TMUS)
- Variance du Taux d'Utilisation des Liens (VTUL)
- Variance du Taux d'Utilisation des Serveurs (VTUS)
- Taux d'acceptance
- Temps de convergence

ÉTS Le génie pour l'industrie

RESULTATS

RESULTATS

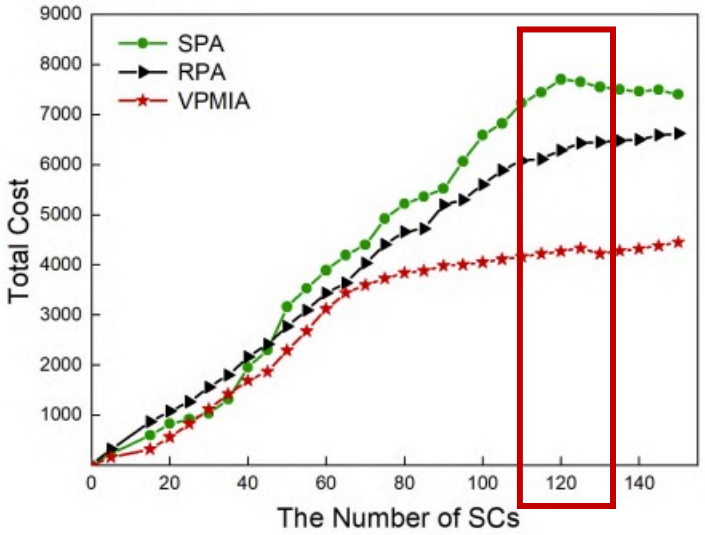


FIGURE 4. Total cost.

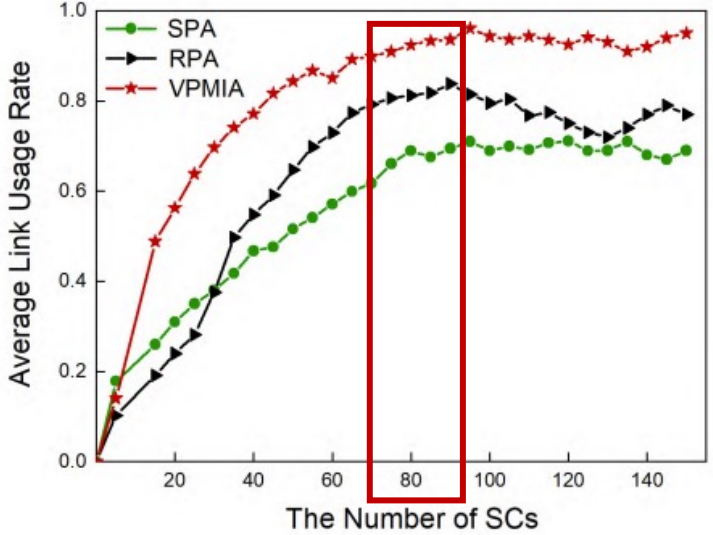


FIGURE 5. Average link usage rate.

- Nombres de SCs = 120 :

SPA 31% supérieur au VPMIA (en Coût)

RPA 22% supérieur au VPMIA (en Coût)

- Nombres de SCs = 80 :

SPA 51% inférieur au VPMIA (en TMUL)

RPA 27% inférieur au VPMIA (en TMUL)

TMUL = Taux Moyen d'Utilisation des Liens

RESULTATS

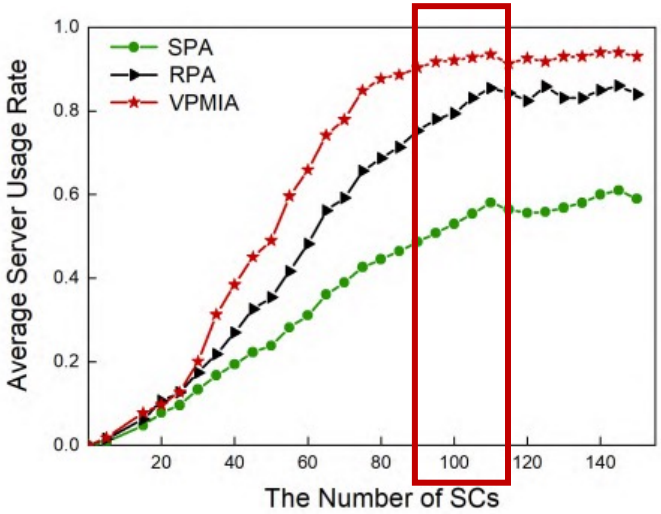


FIGURE 6. Average server usage rate.

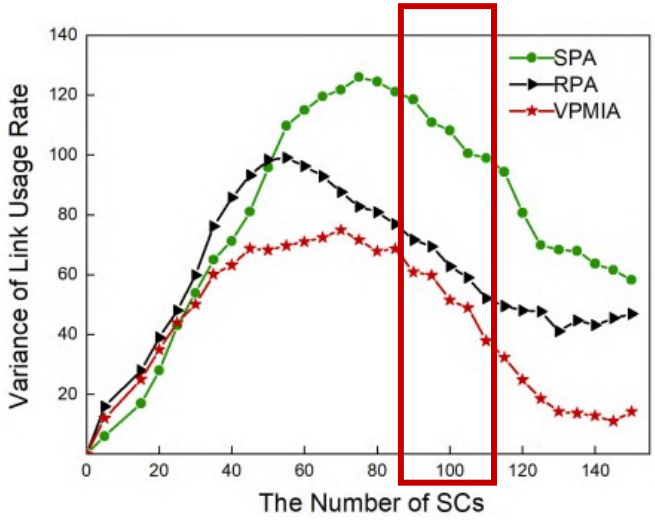


FIGURE 7. Variance of link usage rate.

- Nombres de SCs = 100 :

SPA 47% inférieur au VPMIA (en TMUS)

RPA 16% inférieur au VPMIA (en TMUS)

TMUS = Taux Moyen d'Utilisation des Serveurs

- Nombres de SCs = 100 :

SPA 61% supérieur au VPMIA (en VTUL)

RPA 73% supérieur au VPMIA (en VTUL)

VTUL = Variance du Taux d'Utilisation des Liens

RESULTATS

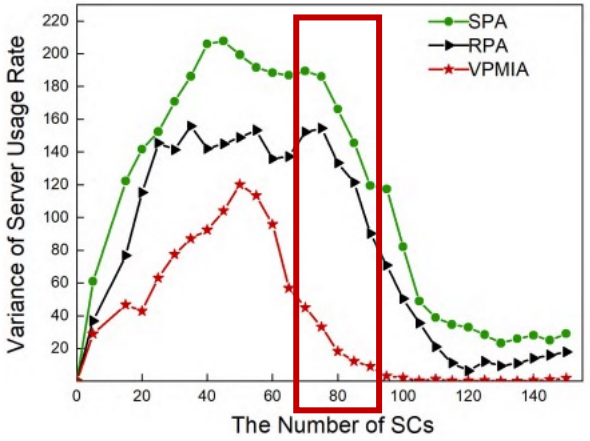


FIGURE 8. Variance of server usage rate.

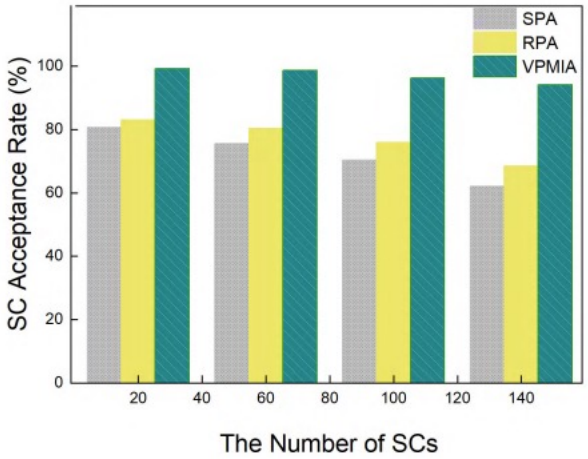


FIGURE 9. Acceptance rate.

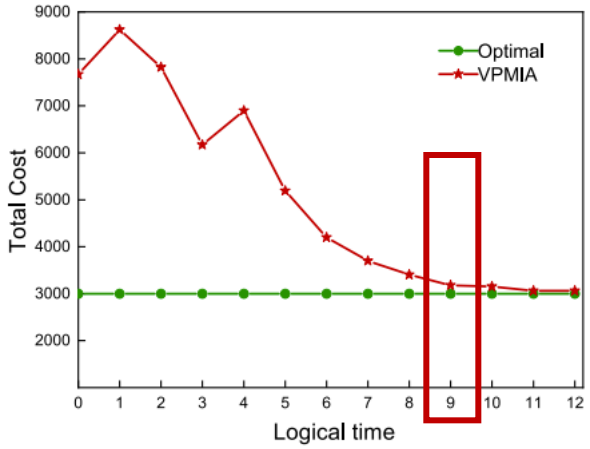


FIGURE 10. Convergence time.

- Nombres de SCs = 80 :

SPA 81% supérieur au VPMIA (en VTUS)

RPA 72% supérieur au VPMIA (en VTUS)

VTUS = Variance du Taux d'Utilisation des Serveurs

- Convergence et solution quasi-optimale pour VPMIA au temps logique 9

LIMITES
&
TRAVAUX FUTURS

LIMITES

- VPMI est très avantageux uniquement lorsque le nombre de SCs est élevé
- VPMI, au début, a une faible efficience car les schémas sont choisis au hasard
- Les résultats ont été obtenus à partir d'une simulation et non en conditions réelles

TRAVAUX FUTURS

- Explorer et comparer l'efficacité des techniques de machine learning (deep learning et reinforcement learning) pour le placement des VNF.
- Considérer la résilience de l'intégration de la chaîne de services avec le concept de découpage en tranches, par exemple.

ÉTS
Le génie pour l'industrie

MERCI !

ÉTS
Le génie pour l'industrie

REFERENCES DE L'ARTICLE

He, W., Guo, S., Liang, Y., & Qiu, X. (2019). Markov Approximation Method for Optimal Service Orchestration in IoT Network. IEEE Access, 7, 49538–49548. https://doi.org/10.1109/access.2019.2910807