Reducing the Service Function Chain Backup Cost Over the Edge and Cloud by a Self-Adapting Scheme

Xiaojun Shang , Yaodong Huang , Zhenhua Liu, and Yuanyuan Yang , Fellow, IEEE

Abstract

Emerging virtual network functions (VNFs) bring new opportunities to network services on the edge within customers’ premises. Network services are realized by chained up VNFs, which are called service function chains (SFCs). These services are deployed on commercial edge servers for higher flexibility and scalability. Despite such promises, it is still unclear how to provide highly available and cost-effective SFCs under edge resource limitations and time-varying VNF failures. In this paper, we propose a novel Reliability-aware Adaptive Deployment scheme named RAD to efficiently place and back up SFCs over both the edge and the cloud. Specifically, RAD first deploys SFCs to fully utilize edge resources. It then uses both static backups and dynamic ones created on the fly to guarantee the availability under the resource limitation of edge networks. RAD does not assume failure rates of VNFs but instead strives to find the sweet spot between the desired availability of SFCs and the backup cost. Theoretical performance bounds, extensive simulations, and small-scale experiments highlight that RAD provides significantly higher availability with lower backup costs compared with existing baselines.

Keywords: Virtual network functions, edge computing, service function chain, availability.

1 Introduction

The development of Virtual Network Functions (VNFs) transforms traditional middleboxes, such as firewalls, load balancers and proxies, on dedicated hardware into virtual functions on commercial servers, thus introducing more flexibility, scalability and cost effectiveness. Much research has been conducted to facilitate such benefits. With the rapid development of edge computing and 5G networks, there is a growing incentive to deploy VNFs at the edge within customer premises for lower latency and better performance. Despite this benefit, separating network functions from their purpose-built hardware may reduce their availability. For example, in many VNF systems, the VNF runs as an instance on a virtual machine (VM) whose resources are managed by the underlying hypervisor. As a result, any failure of the hypervisor may render the VNF running on it unavailable. Worse still, when multiple VNFs are chained together to provide an overall network service, the failure of any VNF on this Service Function Chain (SFC) can render the entire service unavailable. As a result, the availability problem of SFCs is more serious than that of individual VNFs. This thesis proposes a static backup of VNFs as well as a dynamic backup of VNFs, which can be activated in case of failure of the original VNF, to guarantee the availability of the Service Function Chain (SFC) at the smallest possible cost, and expense.[1]

2 Related works

In this paper two algorithms are implemented: a static backup deployment algorithm and a dynamic backup deployment algorithm, and finally showing a visual comparison of each server load.[2].

2.1 Pulp

PuLP is a library for the Python scripting language that enables users to describe mathematical procedures, provides Python objects representing optimisation problems and decision variables, and allows constraints to be expressed in a simple way. In order to make the syntax as simple and intuitive as possible, PuLP focuses on supporting linear and mixed integer models. PuLP can be easily deployed on any system with a Python interpreter, as it does not rely on any other packages. The way it works is divided into four main steps: presenting the optimisation objective, defining the variables, defining the objective function, and defining the constraints.[3]

2.2 Static backup deployment

The deployment of static backups determines the location of each static backup with the aim of minimising the cost of backup without violating the resource constraints of any edge server. The optimisation algorithm is transformed into a shaping linear programming problem by setting constraints. Use LP Solver to solve the static deployment problem.[4]

2.3 Dynamic backup deployment

Second, we use the dynamic backup deployment method, which creates dynamic backups whenever the VNF or its static backups fail. Each dynamic backup is placed on an edge server using an online algorithm. Online optimisation has recently been applied to edge computing and cloud computing. The algorithm is used to balance the load.

This is the main cause of VNF failure. Dynamic backups are released when both the original VNF and static backups are restored. However, if the current availability of an SFC (previous working time/total uptime) is not met, the dynamic backup of this SFC will not be released, thus enhancing the availability of this SFC. Since most VNFs recover after a short period of time and release the corresponding dynamic backups, the resource usage of dynamic backups at each moment is relatively small. Therefore, the remaining edge resources are usually sufficient for dynamic backups, thus guaranteeing the required availability of the SFC.[5]

2.4 Online algorithm for dynamic backup deployment

Dynamic backup deployments are terminated when the current edge resources are insufficient for the upcoming dynamic backups. We then apply an SFC backup tuning method that moves the backup of the lowest cost SFC from the edge to the cloud. The reliability of this SFC is then guaranteed by the cloud and its static and dynamic backups at the edge are released. The online algorithm then continues. As the failure rate of the VNF increases, the scheme adaptively moves more SFCs to the cloud to further guarantee availability. When the failure rate decreases, the solution adaptively backs up more SFCs at the edge to reduce the cost of cloud backup.[6]

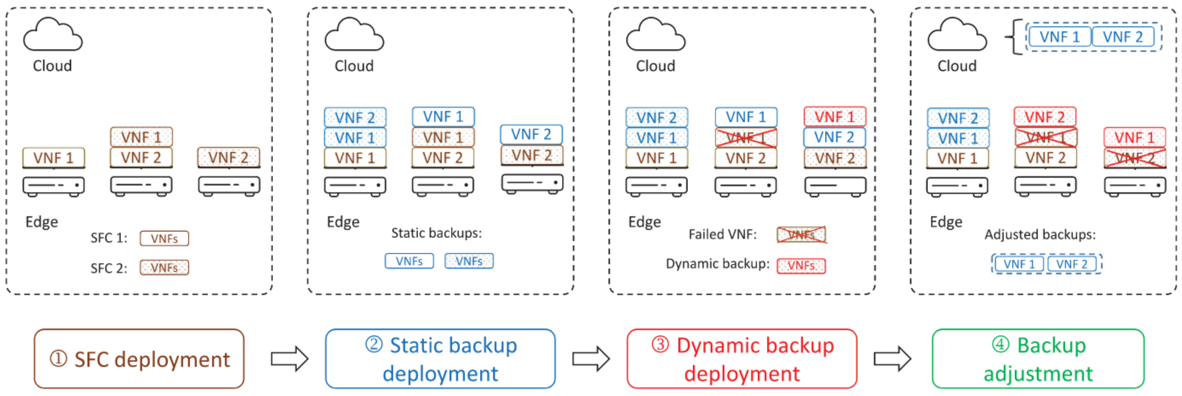

Figure 1: The demonstration of RAD. In Step 1, RAD deploys SFCs in brown to edge servers for the maximal edge resource utilization. VNFs with different patterns belong to different SFCs. In Step 2, RAD deploys each VNF at the edge a static backup in blue for the lowest backup cost. Any SFC or static backup that cannot be deployed at the edge due to resource constraints will be placed in the cloud instead. In Step 3, whenever a VNF fails (marked by a red cross), RAD deploys a dynamic backup in red for the failed VNF and releases it if the VNF recovers. If edge resources are not sufficient for dynamic backups, RAD goes to Step 4 and adjusts static backups of some SFCs to the cloud, thus releasing more edge resources.

3 Implementation details

3.1 Implement thought

Since there is no reference to any relevant source code, We need to analysis and design the data sets one by one first. Because before implementing the algorithm, We need some data as input. And this is the most difficult and important part. We have to find the connections between the notations and transform the math notations into three dimension arrays or tuples.

We use Figure 1 as a brief demonstration of SAB. In the first step, we propose a static backup deployment method, deploying a static backup for each VNF. We do not consider the case where some VNFs of SFC are backed up at the edge and others are backed up in the cloud. The reason is that in our application scenario, the latency between the edge and the cloud is greater than the latency between the edge servers, especially in the emerging 5G environment. Partial backup of SFCs in the cloud may cause service flows to move up and down multiple times between the original VNF at the edge and the backup in the cloud. This process can incur significant latency and add significant traffic to the network.[7] The static backup deployment determines the location of each static backup with the aim of minimising backup costs without violating the resource constraints of any edge servers. The detailed deployment methodology is described in Section 2. The first step of static backup deployment is a prerequisite for the second step, namely dynamic backup deployment and SFC backup tuning. It provides a static backup for each VNF on the edge and allows dynamic backups to be created in the event of a failure of the original VNF. As shown in Figure 1.

The demonstration of RAD. In Step 1, RAD deploys SFCs in brown to edge servers for the maximal edge resource utilization. VNFs with different patterns belong to different SFCs. In Step 2, RAD deploys each VNF at the edge a static backup in blue for the lowest backup cost. Any SFC or static backup that cannot be deployed at the edge due to resource constraints will be placed in the cloud instead. In Step 3, whenever a VNF fails (marked by a red cross), RAD deploys a dynamic backup in red for the failed VNF and releases it if the

VNF recovers. If edge resources are not sufficient for dynamic backups, RAD goes to Step 4 and adjusts static backups of some SFCs to the cloud, thus releasing more edge resources.

3.2 Static Backup Deployment Algorithm

Procedure 1 Static Backup Deployment Algorithm.

```
Input: ${\tilde{x}_{f,i,v}}$ and ${\tilde{y}_f}$ from the LP solver.
Output: Binary output ${x_{f,i,v}}$, ${y_f}$ and threshold $\theta$
 1: Initialize all $\mathrm{x}_{f,i,v} = 0$.
 2: for all $f \in F, i \in I_f$ do
 3: for all $v \in V \backslash o_{f,i}$ do
 4:     if $\tilde{x}_{f,i,v} = \max_{v \in V \backslash o_{f,i}} \{\tilde{x}_{f,i,v}\}$ then
 5:        $x_{f,i,v} \leftarrow 1,$ break.
 6:     end if
 7:    end for
 8: end for
 9: while Constraints are not satisfied do
10:    $\theta \leftarrow \max_{f \in} \{\tilde{y}_f\}$
11:    for all $\tilde{y}_f = \theta$ do
12:        $\tilde{y}_f \leftarrow 0$
13:        $y_f \leftarrow 1$ and corresponding $x_{f,i,v} \leftarrow 0$
14:     end for
15: end while
16: for all $f \in F$ do
17:     if $y_f = 1$ and setting corresponding $x_{f,i,v}$ set to $0$ in line $13$
18:     back to 1 does not violate any constraint then
19:          $y_f \leftarrow 0$ and corresponding $x_{f,i,v} \leftarrow 1$ if
20: end for
```

The details of static backup deployment algorithm is presented in Procedure 1. The algorithm can help us to figure out the minimum cost of placing the static backups. In the algorithm, We try to adjust the output of LP slover to minimzie the total backup cost. We define two binary parameters $\{x_{f,i,v}\}$ and $\{y_f\}$ as output. $\{x_{f,i,v}\}$ is a decision variable whether VNF i of SFC f is deployed on server v. And $\{y_f\}$ is a decision variable whether static backup i of SFC f is deployed on server v. For example, $\{x_{f,i,v} = 1\}$ represent that VNF i of SFC f is deployed on server v.

The algorithm iteratively sets values for the binary variables and transforms scales of [0,1] into 0,1 in lines 4-8, and then decides whether the backup should put into cloud according to the input $\tilde{y}_f$. We updates the threshold value and sets some binary variables to 0 while maintaining feasibility in lines 9-15. Finally, it checks for feasibility again in lines 16-20.

3.3 Dynamic Backup Deployment Algorithm

Procedure 2 Dynamic Backup Deployment Algorithm.

Input: Set of edge servers V, current deployment of VNFs and backups, dynamic backup set K, ε and ρ.

Output: Placement of each dynamic backup $\{x_{k,v}; \forall k \in K, \forall v \in V\}$

```
1: $n \leftarrow 1; \eta_{1,v} \leftarrow 0, \forall v \in V; M_1 \leftarrow \max_{v \in V} \left\{ \frac{b_v}{R_v} \right\}$
2: Whenever a dynamic backup request arrives, over flow $\leftarrow$ True and do lines 3-23.
3: **for all** $v \in V$ **do**
4:     if $\frac{b_v}{R_v} + \sum_{j=1}^{k-1} \frac{\gamma_j}{R_v} \cdot x_{j,v} + \frac{\gamma_k}{R_v} > 1$ then
5:     $V \leftarrow V \backslash v$
6:     **end if**
7: **end for**
8: **if** $V = \emptyset$ **then**
9:     Terminate the algorithm and execute backup adjustment.
10: **end if**
11: **while** over flow $=$ True **do**
12:     over flow $=$ False
13:     **for all** $v \in V$ **do**
14:         $\delta_{n,v}^k \leftarrow \frac{\gamma_k}{R_v \cdot M_n}$
15:     **end for**
16:     Sort servers in $V$ in the increasing order of $\varepsilon^{\eta_{n,v}+\delta_{n,v}^k} - \varepsilon^{\eta_{n,v}}$ to get $V'$, where $\varepsilon \in \left(1, \frac{\rho+1}{\rho}\right], \rho > 1$.
17:     Find the first $v'$ in $V'$ satisfying $v \notin \{o_{k,1}, o_{k,2}\}$.
18:     **if** $\eta_{n,v'} + \delta_{n,v'}^k > \log_\varepsilon \left( \frac{\rho}{\rho-1} |V| \right)$ **then**
19:         $n \leftarrow n + 1; M_n \leftarrow 2 \cdot M_{n-1}; \eta_{n,v} \leftarrow 0, \forall v \in V$; over flow $\leftarrow$ True
20:     **else**
21:         $x_{k,v'} = 1; \eta_{n,v'} \leftarrow \eta_{n,v'} + \delta_{n,v'}^k$
22:     **end if**
23: **endwhile**
```

Dynamic Backup Deployment Algorithm aims to place dynamic backups for virtual network functions (VNFs) in a set of edge servers. The algorithm takes inputs as the set of edge servers, the current deployment of VNFs and backups, a dynamic backup set, and some parameters. It outputs the placement of each dynamic backup.

The algorithm starts by setting some initial values and then waits for a dynamic backup request. When a request arrives, the algorithm checks if adding the backup to any of the servers in the current deployment would cause overflow. If so, the server is removed from the set of edge servers. If there are no servers left, the algorithm terminates and executes backup adjustment. If not, the algorithm calculates some parameters and sorts the remaining servers based on those parameters. It then selects the first server that satisfies some conditions and places the dynamic backup on it. If placing the backup causes overflow, the algorithm repeats the process. Otherwise, it moves on to the next request.

The algorithm uses some variables and parameters, such as b_v and R_v, which represent the backup capacity and available bandwidth of server v, respectively. It also uses γ_k to represent the size of the k-th dynamic backup and $\eta_{n,v}$ to represent the amount of backup already deployed on server v in iteration n. The algorithm also uses $x_{k,v}$ to represent the placement of the k-th dynamic backup on server v. The algorithm sorts servers based on a parameter that involves ε, which is a constant between 1 and $(\rho+1)/\rho$, where ρ is a parameter that determines the trade-off between backup capacity and number of iterations. The algorithm terminates and executes backup adjustment when there are no servers left to place a dynamic backup or when there is an overflow in the network.

3.4 Main contributions

Since there is no reference to any relevant source code, We need to analysis and design the data sets displayed below one by one first.And we have to find the connections between the notations and transform the math notations into three dimension arrays or tuples. Beside, we implement the online algorithm for dynamic backup without pseudocode. Then we display the comparisons of load of each servers between LP slovers method and our algorithm.

Notation	Definition
V	Set of servers on the edge, $V = \{1,2,\dots,v,\dots,\lvert V\rvert\}$
F	Set of SFCs, $F = \{1,2,\dots,f,\dots,\lvert F\rvert\}$
I_f	Set of VNFs of SFC f, $I_f = \{1,2,\dots,i,\dots,\lvert I_f\rvert\}$
$x_{f,i,v} \in \{0,1\}$	Decision variable whether VNF i of SFC f is deployed on server v.
$y_{f,i,v} \in \{0,1\}$	Decision variable whether static backup i of SFC f is deployed on server v.
$z_{k,v} \in \{0,1\}$	Decision variable whether dynamic backup k is deployed on server v.
C	Total Resources on the edge
R_v	Resources on edge server v
a_v	Resource demand on v before deploying static backups
b_v	Resource demand on v before dynamic backups
β_f	Resource demand of SFC f
$\beta_{f,i}$	Resource demand of VNF i of SFC f
w_f	Backup cost of SFC f
$o_{f,i}$	Server holding the VNF i of SFC f
$o_{k,1}, o_{k,2}$	Servers holding VNF and static backup of k
K	Set of dynamic backups $K = \{1,2,\dots,k,\dots,\lvert K\rvert\}$
γ_k	Resource demand of the dynamic backup k
W_v	Load on v after deploying $\lvert K\rvert$ dynamic backups

4 Results and analysis

The load profile of each server is derived by analysing the resource usage on the server and comparing the dynamic backup deployment algorithm with the LP optimised deployment. As shown in Figure 2 which shows that our algorithm behave better than LP slover. In the most situations, the load of each server is more lower.

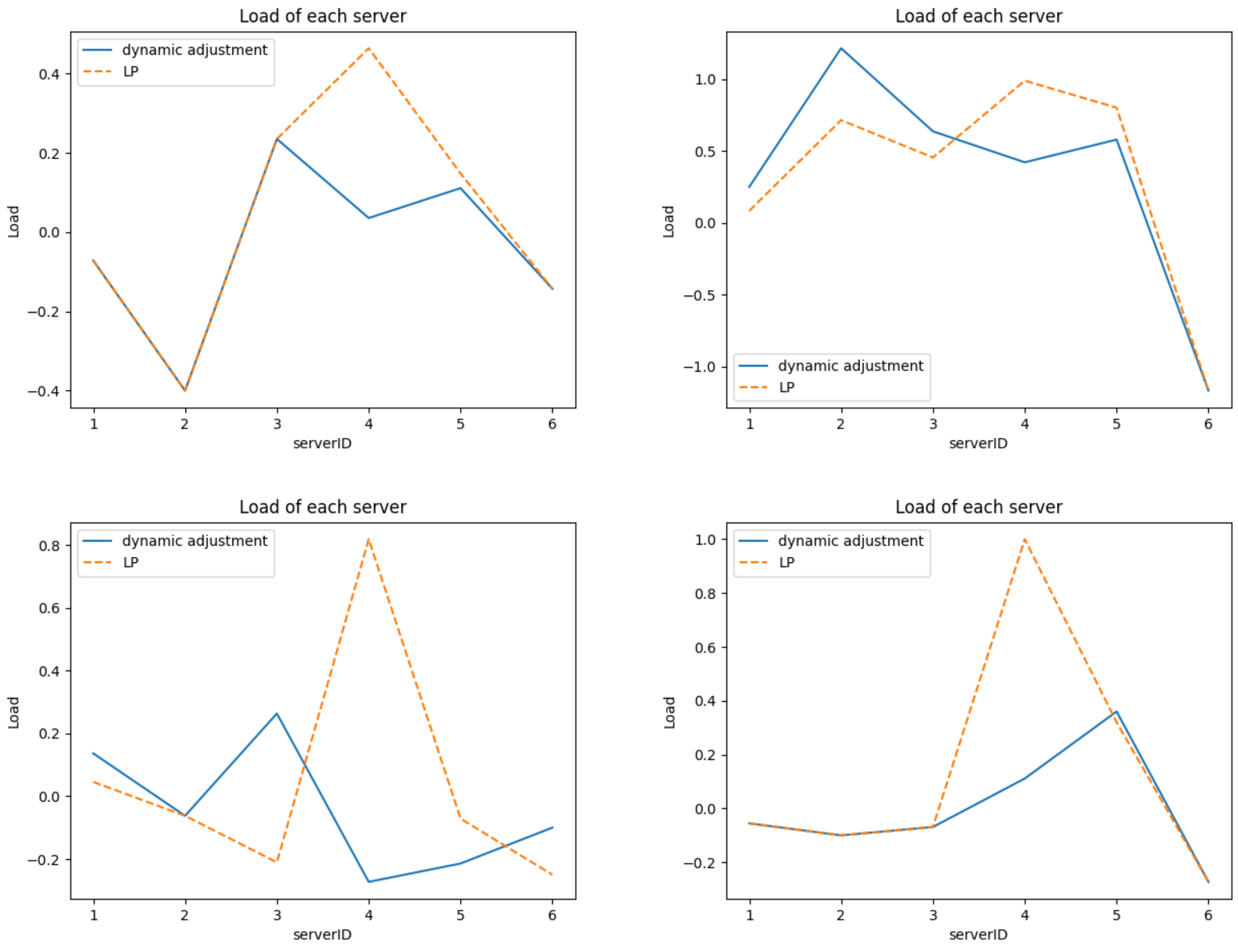

Figure 2: Load of each server

References

[1] ZHANG X, WU C, LI Z, et al. Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization[C]//IEEE INFOCOM 2017-IEEE Conference on Computer Communications. 2017: 1-9.

[2] LI J, GUO S, LIANG W, et al. SFC-Enabled Reliable Service Provisioning in Mobile Edge Computing via Digital Twins[J]. 2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS), 2022: 311-317.

[3] CLARK J D, et al. Pulp technology and treatment for paper.[J]. Pulp technology and treatment for paper., 1985.

[4] FAN J, JIANG M, ROTTENSTREICH O, et al. A framework for provisioning availability of NFV in data center networks[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2246-2259.

[5] HUANG H, GUO S. Proactive failure recovery for NFV in distributed edge computing[J]. IEEE Communications Magazine, 2019, 57(5): 131-137.

[6] SHANG X, LI Z, YANG Y. Placement of highly available virtual network functions through local rerouting[C]//2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). 2018: 80-88.

[7] YALA L, FRANGOUDIS P A, KSENTINI A. Latency and availability driven VNF placement in a MEC-NFV environment[C]//2018 IEEE Global Communications Conference (GLOBECOM). 2018: 1-7.