Service Function Protection in Network Function Virtualization

Service Function Protection In Network Function Virtualization

by

Chengzong Peng

Under the Direction of Xiaojun Cao, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2022

ABSTRACT

In the next generation of networking paradigms, network services are expected to be provided with low latency, high reliability, and high connection density. With network function virtualization (NFV), these network services can be delivered via software-based modules known as service functions. While NFV enables greater flexibility and efficiency in delivering service functions, it poses challenges for network robustness and reliability. In order to maintain network reliability and prevent service interruptions caused by network failures, it is essential to allocate backup resources properly. In this dissertation, we investigate service function protection in NFV and develop efficient methods for allocating backup resources.

First, we discuss how to protect NFV services from service function instance failures in unique function networks. Second, we investigate how to ensure that NFV services are effectively protected from service function instance failure in multiple function networks. Third, we investigate how to adequately protect NFV services from service function forwarder failure in cloud networks. Our fourth study explores how to protect NFV services against service function forwarder failure in edge networks. The fifth part presents an efficient method for protecting NFV services against service function chain failures in the edge networks. We analyze and discuss our proposed protection mechanisms and conduct extensive simulations to assess their effectiveness. Simulation results demonstrate that the proposed protection mechanisms are effective and efficient under various conditions. Lastly, we suggest two possible research directions for future investigation.

INDEX WORDS: Network function virtualization, Software-defined networking, Service function chaining and embedding, Network reliability, Backup resource allocation, Approximation algorithm

Service Function Protection In Network Function Virtualization

by

Chengzong Peng

Committee Chair: Xiaojun Cao

Committee: Anu Bourgeois

Yanqing Zhang

Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2022

DEDICATION

This dissertation is dedicated to my parents, Hao Peng and Li Hua, for their endless support and love during my doctoral studies. Without their love and encouragement, I could not have finished my Ph.D. dissertation.

ACKNOWLEDGMENTS

This dissertation summarizes my doctoral studies at Georgia State University in the United States. Over the past four years, pursuing my Ph.D. degree has been a truly life-changing experience for me. Throughout the process, I would not have been able to complete it without the guidance of my committee members, the help of my research group, and the support of my family and friends.

First, I would like to express my sincere gratitude to my advisor, Dr. Xiaojun Cao, for his guidance, mentorship, and encouragement. Dr. Cao always inspired me when I was frustrated and did not hesitate to provide valuable advice and guidance when I was trapped in research. It would not have been possible for me to find doing research so interesting without Dr. Cao's inspiration. Working under Dr. Cao's supervision gave me invaluable experience that helped me develop my academic and professional skills.

As members of my dissertation committee, I would like to express my gratitude to Dr. Anu Bourgeosis, Dr. Yanqing Zhang, and Dr. Yichuan Zhao for their valuable comments on improving my dissertation. It was a great honor to have them as members of my dissertation committee, and it was a pleasure to work with them.

I want to thank our professors and staff in the Department of Computer Science, especially Dr. Yingshu Li, Dr. Zhipeng Cai, Dr. Wei Li, Dr. Raj Sunderraman, and Dr. Alex Zelikovsky, for their great help during my Ph.D. program. In addition, I would like to thank Ms. Tammie Dudley, Mr. Jamie Hayes, Ms. Tyra Robinson, and Mr. Henry Louis for their

patience and help in making my Ph.D. study and teaching in our department smoother and more accessible.

I want to express my gratitude to Dr. Ling Tian and Dr. Yimin Zhou. I regard Dr. Ling Tian and Dr. Yimin Zhou as role models who initially inspired and motivated me to apply for the Ph.D. program. Additionally, I am thankful for the help and guidance from Dr. Guangchun Luo during my doctoral studies.

In addition, I would like to express my appreciation for my research team and colleagues in the department. Special thanks to my research group colleague, Dr. Danyang Zheng, who supported and encouraged me. Also, thanks to Dr. Xueting Liao, Dr. Maryam Jalalitabar, Dr. Evrim Guler, and Yihan Zhong. Thanks to my friends Dr. Xu Zheng, Dr. Ke Yan, Dr. Yiwen Tang, Xuebin Wu, Yutong Gao, Junzhi Wen, Kenan Zhang, Yibing Xie, Zuobin Xiong, Honghui Xu, and Jishen Yang, who helped me with my studies and life these four years.

Lastly, I wish to express my gratitude to my parents, grandparents, and Ms. Yi Shi for their support, encouragement, and care throughout my life and while studying in the United States. Without their constant support and encouragement, I would never have achieved anything.

My sincere gratitude to everyone who contributed to making this dissertation possible. I apologize for not mentioning them individually.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

LIST OF FIGURES

LIST OF ABBREVIATIONS

- NV - Network virtualization
- CPU - Central processing unit
- SDN - Software-defined networking
- API - Application programming interfaces
- FW - Firewall
- LB - Load balancer
- IDS - Intrusion-detection system
- CAPEX - Capital expenditures
- OPEX - Operational expenses
- NFV - Network function virtualization
- ETSI - European telecommunications standards institute
- NFVI - NFV infrastructure
- VNF - Virtual network function
- MANO - NFV management and orchestration
- SF - Service function
- NSR - NFV/Network service request
- SFC - Service function chain

- PN - Physical network
- SFP - Service function path
- SFF - Service function forwarder
- SFCE - Service function chaining and embedding

CHAPTER 1
INTRODUCTION

1.1 Network Virtualization

Over the past few decades, the Internet has been a tremendous success, driving the development of various distributed applications and Web technologies. However, its rapid expansion has encountered the most significant obstacles associated with its widespread adoption. Due to its inherent multiple providers, altering an existing Internet architecture or designing a new framework requires consensus among competitors. As a result, modifications to the Internet architecture are restricted to simple incremental upgrades, and the implementation of new network technologies is challenging [1, 2, 3].

Network virtualization (NV) has been introduced as a critical role in the next-generation networking paradigm in response to network ossification [4]. With network virtualization, network resources that were traditionally hosted in hardware can be abstracted into the software. Several different computer systems can run as virtual machines on the same physical machine, which provides the corresponding computing resources such as a central processing unit (CPU) and memory. This indicates that clients or applications with different requirements can share the same hardware, whereas physical computing resources can be utilized efficiently. As a result, network virtualization releases the inherent limitations of the existing Internet and allows heterogeneous network architectures to coexist.

Network virtualization platforms deliver virtualized resources to customers economically and dynamically to enable flexible resource allocation, agile delivery options, and intelligent

service deployment. With network virtualization support, service providers must efficiently manage these resources and intelligently control service delivery. Following that, we will introduce the concept of software-defined networking (SDN).

1.2 Software-Defined Network

In traditional networks, dedicated hardware-based network devices, such as switches and routers, control and forward the traffic flow of dedicated network services [5]. To provide a new network service, network operators must manually program multiple vendor-specific hardware devices. As a result, painful network reliability, scalability, and speed issues appear in traditional networks.

The SDN paradigm provides a flexible and deployable network architecture with network virtualization by decoupling the control plane from the underlying data plane. With the separation of the control and data plane, SDN deploys the control plane, which determines where and how to send traffic, as a software-based centralized controller, while leaving the data plane, which actually forwards the traffic in hardware [6]. Typically, in large-scale SDNs, the control plane can directly operate over the state in the hardware devices of the data plane through well-defined application programming interfaces (APIs).

Fig. 1.1 introduces the basic architecture of SDN, which includes three planes and communication interfaces: i) application plane, ii) control plane, and iii) data plane. The application plane runs SDN-based business applications on network devices, allowing for relevant requested modifications. The control plane provides logical control policies (e.g.,

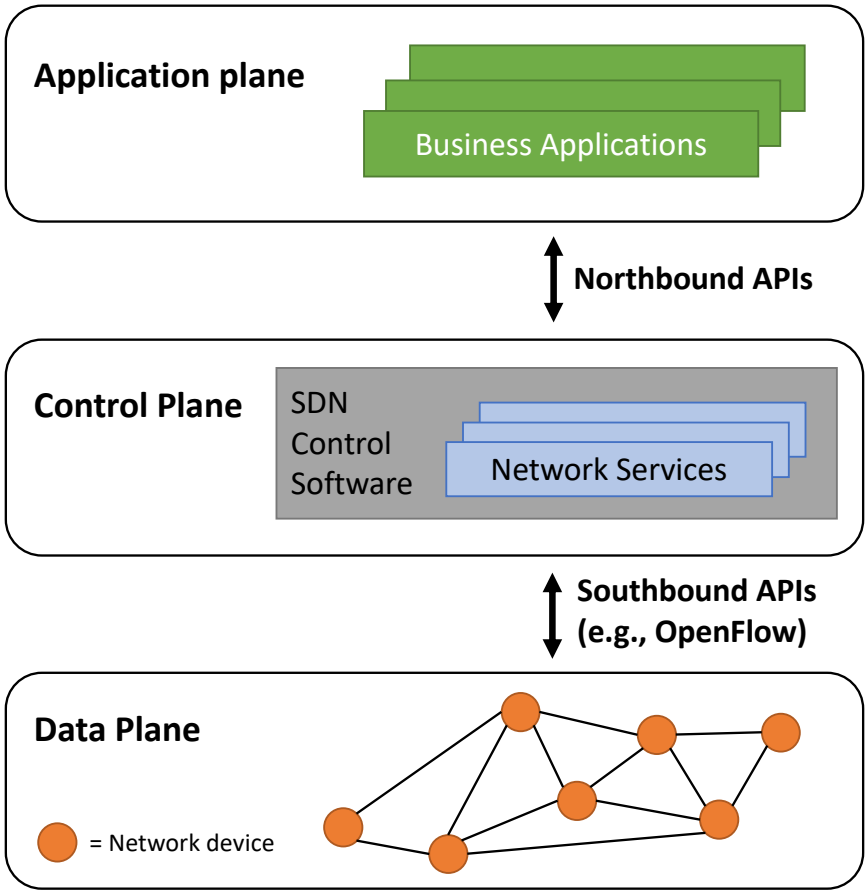


Figure 1.1: Architecture of SDN.

routing schemes) to manage the information collected from the network devices in the data plane. Notably, the control plane has a global view of network conditions, such as resource availability. In the data plane, network devices are responsible for forwarding data when programmable flow tables can be configured dynamically by the control plane. Northbound APIs provide abstractions for application plane development applications, while southbound APIs (e.g., OpenFlow [7]) define the instruction set for communication between the control and data planes. As a result, the SDN architecture makes it easy to implement protocols or policies supporting new network services or applications.

1.3 Network Function Virtualization

Network functions, such as firewalls (FWs), load balancer (LB), and intrusion-detection systems (IDS), are generally implemented in proprietary middleboxes. Managing, deploying, and maintaining these proprietary middleboxes introduce high capital expenditures (CAPEX) and operational expenses (OPEX). To reduce the CAPEX and OPEX, network function virtualization (NFV) is introduced, which the architecture framework comprises a standard developed by the European telecommunications standards institute (ETSI) [8].

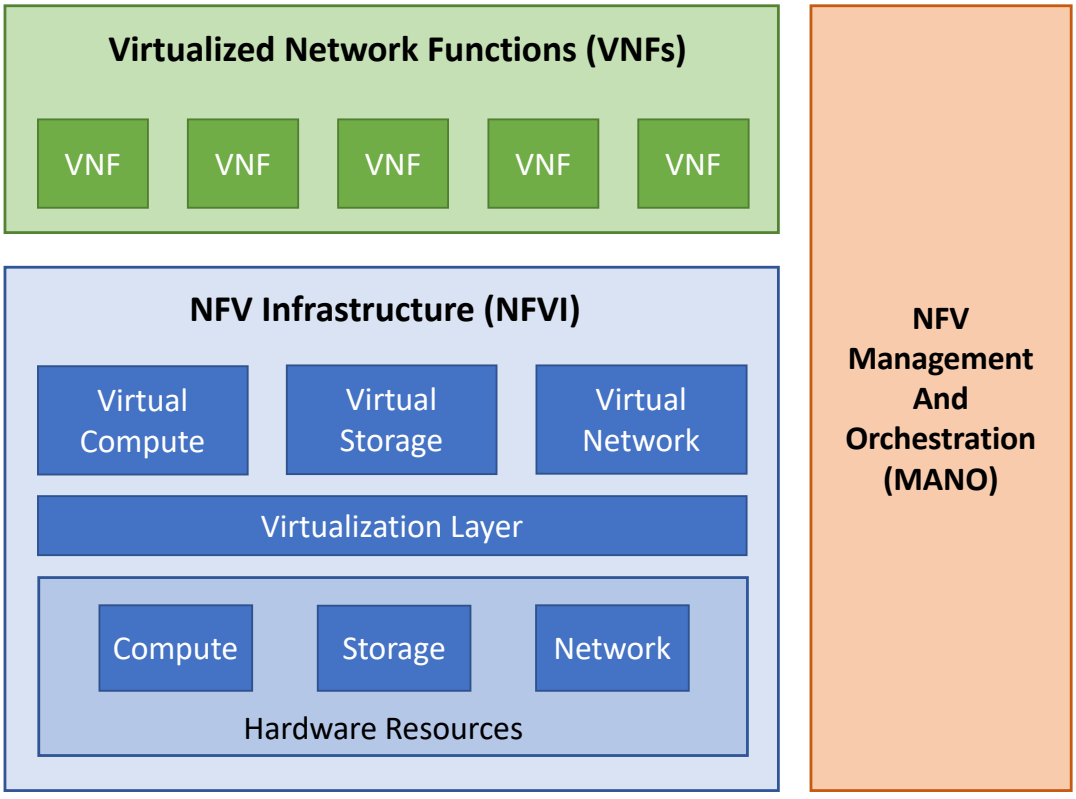


Figure 1.2: NFV architecture framework.

Fig. 1.2 shows the NFV architecture framework, which is composed of three components, the NFV infrastructure (NFVI), virtual network function (VNF), and the NFV management

and orchestration (MANO). The NFVI is the physical infrastructure used to create and maintain the VNFs, which are the counterparts of the physical infrastructure of a systems virtualization environment. Specifically, they are composed of physical hardware resources like computing, storage, and networking resources and are the physical platforms on which virtual computing, storage, and networking required for the VNFs are provisioned. The MANO is the framework component that is used to create, launch, and maintain the VNFs, and provides coordination between the NVFI and the VNFs. Specifically, this is the administrator's interface to manage the whole NFV environment.

With NFV, network services can be delivered with significant flexibility, scalability, and adaptability by virtualizing the proprietary middleboxes into software-based modules running on commodity servers, named VNFs or service functions (SFs) [9]. An NFV service request (NSR) from the clients consists of a set of SFs and the related networking resources. To accommodate an NSR, the service provider chains the required SFs into a linear structure with the virtual link, called service function chain (SFC), and reserve the corresponding networking resources to embed onto a shared physical network (PN) [10]. The logical path of an SFC is called a service function path (SFP). Forwarding traffic along such an SFP is the responsibility of the service function forwarder (SFF). Note that, in a physical node, a virtual machine that runs a specific SF is called an SF instance. The process of accommodating an NSR in a shared PN is referred to as service function chaining and embedding (SFCE)[11].

1.4 Why Does Protection in NFV Matter?

As NFV becomes more adaptable in different application prototypes, network reliability in NFV has become a significant concern. One of the challenges in NFV is efficiently allocating backup computing resources to meet the different requirements of clients when different network failures occur.

Any network element, whether physical or virtual components, such as a physical node, SF instance, SFF, and SFC, can fail for several reasons, such as human misconfiguration, hardware malfunctions, software crashes, and connection errors [12, 13, 14, 15]. Any network element failures can trigger severe issues for the network and the associated clients. In addition, clients' satisfaction will be significantly reduced and may even cause a potential ripple effect on their assets and lives if the required services suddenly shut down and cannot be recovered in time under the network failure. For example, the failure of any physical node would make all the running SFs hosted unavailable for clients. If any SF instance failure occurs and cannot be recovered promptly, the SFC running this SF may become unavailable, interrupting all network services that use this function. If any SFF failure occurs and does not respond for recovery immediately, multiple SFs that run for different network services become unavailable. The entire network service may become unavailable if any SFC failure occurs and is not recovered immediately.

Meanwhile, the potential occurrence of multiple simultaneous failures of network elements in NFV increases the difficulty of protecting NFV services. Several network elements may be destroyed simultaneously due to extreme natural disasters, or the second failure may

occur before the first failure is recovered. For this reason, backup resource allocation with protection strategies to handle different failure scenarios has become essential in NFV.

1.5 Dissertation Organization

Although NFV has the potential to deliver network services with greater flexibility and efficiency, it can also lead to challenges when it comes to network robustness, one of which is improving network reliability through NFV. A major purpose of this dissertation is to investigate how to protect network services to clients in NFV against a variety of SF failure scenarios. In Chapter 2, we investigate how to protect NFV services from SF instance failure in unique function networks. Chapter 3 discusses how to protect NFV services from SF instance failure in multi-function networks. In Chapter 4, we look at the problem of effectively protecting NFV services against SFF failures in cloud networks. In chapter 5, we continue to investigate how to effectively protect NFV services from SFF failures at the edge networks. Chapter 6 explores the problem of protecting NFV services from any SFC failure in edge networks. In Chapter 7, we conclude this dissertation works.

CHAPTER 2
SERVICE FUNCTION INSTANCE FAILURE AND PROTECTION IN UFNS

2.1 Motivation

Network function virtualization (NFV) transforms the implementation of the dedicated proprietary hardware into software appliances, called virtual network functions (VNFs) [16, 17, 18]. VNFs can be flexibly installed on or removed from high volume servers in the substrate network (SN) [19]. In NFV, a network request from a customer includes a set of service functions (or VNFs) and corresponding resource demands, such as bandwidth, CPU and storage [20]. To satisfy the request from a customer, the service provider chains a required VNFs into a virtual catenary structure with VNF links, which is called service function chain (SFC) [21]. At the same time, enough networking resource is reserved for embedding the SFC onto the shared SN [22]. To satisfy the increasing bandwidth demands, the elastic optical network (EON) is widely adopted [23]. In an EON, FlexGrid fibers are used to provide the bandwidth to connect the substrate nodes [24]. Thanks to the virtualization techniques, VNFs from different network requests may be deployed onto the same SN node [25]. As a result, one physical node or fiber link failure may affect multiple network requests [26]. To recover from the network failure (e.g., node/link failure), traditional network survivability schemes may detour the traffic or find backups to comply with the service level agreement (SLA) constraints and service function requirements [27].

Recently, how to effectively handle an VNF failure has attracted the attention of the networking research community [28, 29]. While the traditional node/link failure alters the

network topology and connectivity, an VNF failure does not disrupt the passing-by traffic flow but affects the customer's business process [30]. Such VNF failures may disrupt the output of the data flows within the SFC, or break the data integrity, confidentiality or availability requirements of the customer. Taking a VNF of IDS as an example, its failure will compromise the network defense for the organization. In the literature, some backup and detouring techniques are proposed to handle VNF failures under some specific scenarios [31, 32]. In [31], the authors try to enhance the service reliability with a flow and SFC parallelism strategy. The work in [32] investigated the reliable VNF service provisioning in the multi-access edge computing system by providing primary and backup SF instances.

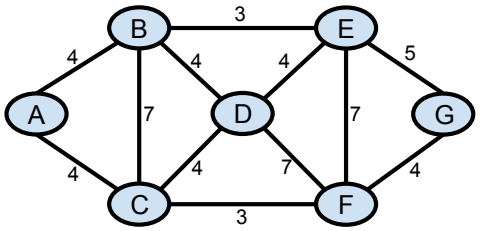
In this chapter, we study how to select a minimum set of SN nodes to protect VNF failures in an EON. We assume the EON is an unique function network (UFN), whereas each SN node can only host one unique VNF. All VNF computing load on an SN node requires a backup SN node, namely the VNF backup (VB) node. For example, if the VNFs on an SN node i fail, the SN node $vb_i \neq i$ will take over and resume the VNF computing load at i. Meanwhile, the latency between i and vb_i must meet a certain QoS constraint. We assume that at one time, only VNFs from single SN node may fail [33, 34]. Hence, VNFs from multiple SN nodes can share a VB node. We define this problem as VNF failure protection cover (VFPC), and propose an efficient heuristic algorithm, namely, protection cover list-based VNF protection (PCL-VP), which guarantees an $ln(|N|) + 1$-approximation boundary, where $|N|$ represents the size of the SN.

The rest of this chapter is organized as follows. In Section 2.2, we mathematically

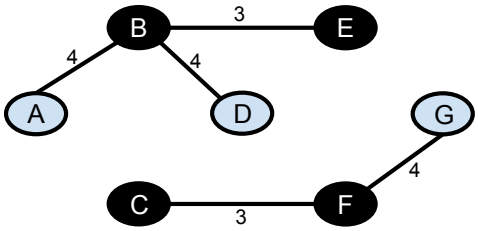
formulate VFPC and prove the NP-hardness of the VFPC problem. Section 2.3 presents the protection cover list-based VNF protection (PCL-VP) algorithm. In Section 2.4, we analyze and evaluate our proposed schemes. Finally, we conclude this chapter in Section 2.5.

2.2 VNF Failure Protection Cover (VFPC)

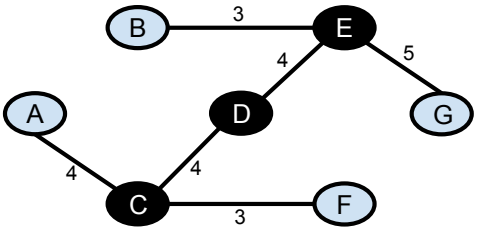
In this section, we define the problem of VNF failure protection cover (VFPC).



(a) Substrate optical network.



(b) B protects A, D, E; E protects B; F protects C, G; C protects F.



(c) E protects B, D, G; D protects E; C protects A, F; D protects C.

Figure 2.1: Example of VFPC.

Table 2.1: VNF computing load and computing capacity.

Physical Node	A	B	C	D	E	F	G
VNF Computing Load	20	35	30	40	30	40	25
Computing Capacity	30	40	50	35	40	40	20

We use the example in Fig. 2.1 and Table 2.1 to illustrate the concept of VFPC. Fig. 2.1a is the topology of an SN, where the number nearby each fiber link represents the latency. Table 2.1 lists the amount of computational loads (to be protected) and available computing

capacity for each physical node[1]. For the VNF load on an SN node i, VFPC needs to identify another SN node vb_i to provide protection (or work as a backup node). At the same time, vb_i must satisfy the following two constraints: (i) the remaining computing capacity at vb_i is no less than the VNF computing load at i; and (ii) the latency from i to vb_i cannot exceed the QoS constraint. For example, in Fig. 2.1, the VNF on the substrate node F has a load of 40 to be protected. If we assume the QoS constraint is 5, then only nodes C and G can be the backup candidates satisfying the QoS constraint for node F. As node G does not have enough computing capacity to protect node F, only node C can be the backup node for node F. Similarly, node F can be the backup node for node C because F has enough capacity to cover C's computing load.

Now, for all the VNFs in the SN, the objective of VFPC is to identify a minimum set of SN nodes such that each VNF computing load has a legit backup node that satisfies the two constraints as mentioned above. Fig. 2.1b shows an approach by selecting the minimum number of physical nodes after applying the minimum dominating set (MDS) first strategy, then identifying backup nodes to protect the nodes in MDS. For example, node B and F can be the two selected MDS nodes to protect all other VNF computing load (i.e., excluding B and F). In specific, node B can protect nodes A, D and E while node F covers nodes C and G. Next, we can identify two nodes to protect nodes B and F. Ideally, if nodes B and F could protect each other, then VFPC would only need two nodes (i.e., B and F) to protect all the NFV computing load. However, the latency between node B and F is beyond the QoS constraint. Hence, we can only find node E to protect node B, and node

[1]We use the term of physical node and substrate node (SN node) interchangeably.

C to protect node F. In the end, as shown in Fig. 2.1b, we need four nodes (i.e., B, E, C and F) to ensure all the NFV computing load is protected. In fact, Fig. 2.1c shows another approach that only requires three selected physical nodes (i.e., C, D, E) to protect all the NFV computing load within the whole SN.

2.2.1 Substrate Network Model

We model the SN with existing VNF computing load as an undirected graph $G = (N, L)$, where N and L denote the set of physical nodes and fiber links, respectively. Each physical node $i \in N$ is associated with a certain number of available computing capacity δ_i and VNF computing load CL_i. For each fiber link $l_{m,n} \in L$, where m and n ($\forall m, n \in N$) are two different physical nodes, its latency is $d_{l_{m,n}}$. The smallest latency between two physical nodes is denoted as $d_{m,n}$, which equals the latency of the shortest path (between the two physical nodes).

We define a VNF backup (VB) node candidate vb_i as a backup node of i, where (i) the latency between i and vb_i is within the QoS constraints, (ii) node vb_i has enough computing capacity to establish VNF instances to protect the VNFs on node i, and (iii) vb_i is a different physical node from i, which is also called offsite backup constraint. For an SN with an existing VNF computing load, the set of VB nodes is referred to as VNF backup cover (VBC).

2.2.2 Problem Formulation

Given an SN with the existing VNF computing load, we study how to select the minimum set of VB nodes to protect all the existing VNFs while satisfying i) the QoS latency constraint,

ii) the computing constraint, and iii) the offsite backup constraint. We name this problem as *VNF failure protection cover* (VFPC).

QoS latency constraint: To satisfy the quality of service (QoS) and service level agreement (SLA) constraint, we define a latency threshold θ. This constraint ensures that the latency from node i to its backup node vb_i: $d_{i,vb_i} \leq \theta$.

VNF computing constraint: If a node does not have enough computing capacity, this node cannot be the backup candidate. Hence, we have $CL_i \leq \delta_{vb_i}$.

Offsite backup constraint: All VNF computing load on a particular SN node i has to find a backup node that is not i. Since assuming only a single SN node failure (at one time) [33, 34], one VB node can backup VNF computing from multiple SN nodes. And all the VNF computing in the substrate network has to be protected.

Let $V_i^j = 1$ represent that if the VNF computing load on node i is protected by node j, 0 otherwise. Hence, the object function of VFPC can be written as Eq. (2.1).

$$\min \sum_{\forall i,j \in N} V_i^j \tag{2.1}$$

Subject to:

$$\text{if} \quad V_i^j = 1, \text{then} \quad vb_i = j \tag{2.2}$$

$$d_{i,vb_i} \leq \theta \tag{2.3}$$

$$CL_{i,vb_i} \leq \delta_{vb_i} \tag{2.4}$$

$$\sum_{j \in N, \neq i} V_i^j = 1, \forall i \in N \tag{2.5}$$

To convert Eq. (2.2) into the linear format, we introduce a large constant M and replace Eq. (2.2) with the following constraint.

$$V_i^j \cdot (M+1) - M \leq vb_i - j + 1 \leq M + 1 - V_i^j \cdot M \tag{2.6}$$

Note that when $V_i^j = 1$, Eq. (2.6) is equivalent to the following two equations.

$$1 \leq vb_i - j + 1 \leq 1$$

$$vb_i = j$$

Otherwise, when $V_i^j = 0$, Eq. (2.6) does not constrain anything. Hence, the mathematical formulation of VFPC includes Eq. (2.1), (2.3), (2.4), (2.5) and (2.6). Eq. (2.3) is the QoS latency constraint and Eq. (2.4) specifies the VNF computing constraint. Eq. (2.5) ensures that the VNF computing load of each physical node is protected by another physical node.

Theorem 2.2.1. *The VFPC problem is NP-hard.*

Proof. To demonstrate the NP-hardness of the VFPC problem, we prove that the minimum dominating set (MDS) problem, a well-known NP-hardness problem [35], is a sub-problem of the VFPC problem. By omitting the constraints above, a special scenario of the VFPC problem can be described as: given a set of elements and a set of non-empty subsets, one need select a minimum number of nodes from the subsets to cover all the elements, which is the same as the MDS problem. Thus, the VFPC problem is NP-hard. □

2.3 Protection Cover List-based VNF Protection (PCL-VP)

In this section, we propose the protection cover list-based VNF protection (PCL-VP) algorithm to optimize the VFPC problem. PCL-VP is built on two proposed techniques: latency-based graph (LBG) and pairwise protection cover first (PPCF).

2.3.1 Latency-based Graph (LBG)

As described in Section II, all the VB nodes need to satisfy the latency and computing constraints. Hence, we propose the process in Algorithm 1 to add auxiliary links between two SN nodes if these two nodes can be connected with enough fiber-subcarriers while satisfying the latency constraint. In specific, Line 1 initializes an undirected graph with all physical nodes and an empty set of links. Line 2-7 add links to the generated new graph G' according to the QoS constraint.

Algorithm 1 Latency-based Graph (LBG)

1: **Input:** G, θ;
2: **Output:** G';
3: Generate an undirected graph $G' = (N, L')$, such that N represents the set of physical nodes that are the same as they are in G and initialize L' as an empty set;
4: **for** $\forall m, n \in N$ **do**
5: **if** $d_{m,n} \leq \theta$ **then**
6: $L' = L \cup l_{m,n}$;
7: update L';
8: **end if**
9: **end for**
10: **return** G';

2.3.2 Pairwise Protection Cover First (PPCF)

In G', let Nei_n represent the set of physical nodes that can be protected by n, which is the protection cover list (PCL). Intuitively, selecting n that has a larger $|Nei_n|$ to be the backup node can help minimize the VBC set. However, such greedy selection of n may have limited choices to protect the nodes in VBC. For example, in the worst case, each of such n will need an additional node added into VBC to protect n. Recall the example in Fig. 2.1b, we add node B and F to VBC first. Then we need two other nodes E and C to protect B and F, respectively. As a result, the VBC set may not be optimized. This is primarily due to the fact that the selection process only considers the PCL (i.e., Nei_n) of one node. Here, we propose to simultaneously consider PCLs from multiple nodes to facilitate optimizing the VBC set. Accordingly, we propose a pairwise protection cover first (PPCF) process as shown in Algorithm 2. The basic idea of PPCF is to select a pair of VB nodes that own larger common PCL (i.e., max $|Nei_n \cup Nei_m|$, $\forall m, n \in N$) and add a link to connect the pair in G' (i.e., $l_{m,n} \in G'$). More specifically, Line 2 creates/updates the PCL of each VB node, while Line 3-7 update the pair of VB nodes that has the largest common PCL.

Algorithm 2 Pairwise Protection Cover First (PPCF)

```
1: Input: G';
2: Output: m and n;
3: Create empty VB nodes m and n;
4: Create/Update the protection cover list (PCL) for each VB node;
5: for each pair of VB nodes a, b (l_{a,b} ∈ G') do
6:     if Nei_a ∪ Nei_b ≥ Nei_m ∪ Nei_n then
7:         m = a, n = b;
8:     end if
9: end for
10: return m and n;
```

2.3.3 Protection Cover List-based VNF Protection (PCL-VP)

Based on the two techniques above, we propose the protection cover list-based VNF protection (PCL-VP) scheme as shown in Algorithm 3. Line 1-2 initialize the variables for the algorithm. Line 3 transfers the input SN into a latency-based graph G'. Line 4-13 select the VB node(s) into VBC until the SN is protected by the VBC. Line 5-6 update the potential VB nodes for the existing VBC and select a VB node c in Nei_{VBC} that owns the largest PCL. Line 7 employs the PPCF technique to select a pair of VB nodes a and b with the largest common PCL. Line 8-12 compare the PCL of (a,b) and c and add the one that owns larger PCL into VBC.

Algorithm 3 Protection Cover List-based VNF Protection (PCL-VP)

1: **Input:** G, θ;
2: **Output:** VBC;
3: Initialize VBC, *Protected* as two empty sets;
4: Create three empty VB nodes a, b and c;
5: $G' = LBG(G, \theta)$;
6: **while** *Protected* does not equal N **do**
7: Update the PCL for VBC (i.e., Nei_{VBC});
8: Select c from Nei_{VBC}, which has the largest PCL;
9: $(a, b) = PPCF(G')$;
10: **if** $\frac{|Nei_a \cup Nei_b|}{2} \geq |Nei_c|$ **then**
11: Add a, b into VBC and add $\{Nei_a \cup Nei_b\}$ into *Protected*;
12: **else**
13: Add c into VBC and add $\{Nei_c\}$ into *Protected*;
14: **end if**
15: **end while**
16: **return** VBC;

Lemma 2.3.1. *Each iteration of PCL-VP does not violate the offsite backup constraint.*

Proof. In the PCL-VP algorithm, there are only two types of VB nodes that will be added

into the VBC in each iteration, a pair of offsite VB nodes or a VB node that is protected by an existing VBC. For the former case, since the two VB nodes provide the protection for each other, they will not violate the offsite backup constraint. For the latter case, since the single VB is protected by an existing VBC, it does not violate the offsite backup constraint, either. □

Theorem 2.3.2. *The proposed PCL-VP algorithm guarantees an* $ln(|N|)+1$*-approximation, where* $|N|$ *is the number of physical nodes in* G*.*

Proof. Assume that the proposed algorithm requires T iterations to select VB nodes protecting all the VNF computing in the SN. Let η_i denote the number of selected VB nodes in the i_{th} iteration; OPT_i denote the optimal solution of the VFPC problem in the i_{th} iteration; $|\Delta_i|$ represent the number of physical nodes protected by the PCL-VP algorithm in the i_{th} iteration; and $|R_i|$ be the number of unprotected physical nodes in the i_{th} iterations. The relationship among $|R_i|$, $|R_{i+1}|$ and $|\Delta_i|$ can be described as Eq. (2.7). $|R_0|$ (the uncovered physical nodes in 0_{th} iteration) equals to the size of substrate network as shown in Eq. (2.8).

$$|R_i| = |\Delta_i| + |R_{i+1}| \tag{2.7}$$

$$|R_0| = |N| \tag{2.8}$$

In Eq. (2.9), the left fraction denotes the average PCL of each VB node selected by the proposed algorithm in iteration i, while the right fraction represents the average PCL of

each VB node in OPT_i. Since the proposed algorithm selects VB node(s) out of the largest average PCL, Eq. (2.9) holds.

$$\frac{|\Delta_i|}{\eta_i} \geq \frac{|R_i|}{OPT_i}, \forall i \in [0, T] \tag{2.9}$$

Then, one can transform Eq. (2.9) into Eq. (2.10).

$$\eta_i \leq \frac{|\Delta_i|}{|R_i|} \times OPT_i, \forall i \in [0, T] \tag{2.10}$$

Accordingly, based on Eq. (2.7), we have Eq. (2.11).

$$\frac{|\Delta_i|}{|R_i|} \leq \frac{1}{|R_i|} + \frac{1}{|R_i| - 1} + \ldots + \frac{1}{|R_i| - |\Delta_i| + 1}$$

$$\leq \sum_{|R_{i+1}| \leq y \leq |R_i|} \frac{1}{y}, \forall i \in [0, T] \tag{2.11}$$

When accumulating the results of the algorithms in T iterations, we can transform Eq. (2.10) into Eq. (2.12). According to Eq. (2.10), Eq. (2.12) can further be formatted as Eq. (2.13), and the left fraction of Eq. (2.12) is the solution from the proposed PCL-VP.

$$\sum_{i=0}^{T} \eta_i \leq \sum_{i=0}^{T} (\frac{|\Delta_i|}{|R_i|} \times OPT) \tag{2.12}$$

$$\sum_{i=0}^{T} \eta_i \leq \sum_{|R_{i+1}| \leq y \leq |R_i|} \sum_{i=1}^{T} (\frac{1}{y} \times OPT_i)$$

$$\leq \sum_{|R_T| \leq y \leq |R_0|} (\frac{1}{y} \times OPT) \tag{2.13}$$

Based on the property of harmonic series, we can further extend Eq. (2.13) to Eq. (2.14):

$$\sum_{i=1}^{T} \eta_i \leq (ln(|N|) + 1) \times OPT \tag{2.14}$$

Hence, the proposed PCL-VP algorithm can reach the result that is bound by $ln(|N|)+1$-approximation, where $|N|$ is the number of physical nodes in the given substrate network. □

2.3.4 An Illustrative Example of PCL-VP

To demonstrate how PCL-VP works, we take Fig. 2.2 and Table 2.2 as the input to calculate the average PCL value of each VB node in each iteration. Before calculating the average PCL value, the LBG algorithm transfers the SN in Fig. 2.1a to a latency-based graph as shown in Fig. 2.2. Then, in the first iteration, the existing VBC is empty and PCL-VP applies the PPCF technique to calculate PCL for each VB node. PCL-VP calculates the average PCL value for each possible pair of physical nodes and selects the pair of VB nodes with the largest common PCL. In Fig. 2.2, due to the average PCL value for node C and D is higher, the PPCF first selects node C and D. In the second iteration, since the unprotected physical node is G, PCL-VP calculates the average PCL value for each possible VB node and selects a VB node with a larger PCL in Nei_{VBC}, which is node E or F. Hence, the minimum VBC to protect the whole SN is the set of $\{C, D, E\}$ or $\{C, D, F\}$.

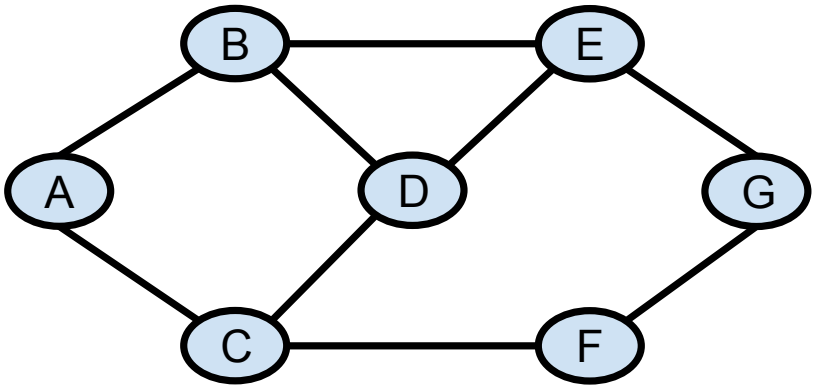


Figure 2.2: Example of LBG.

Table 2.2: PCL and VBC.

Physical Node	A	B	C	D	E	F	G
1_{st} iteration	$\frac{4}{2}$	$\frac{5}{2}$	$\mathbf{\frac{6}{2}}$	$\mathbf{\frac{6}{2}}$	$\frac{5}{2}$	$\frac{5}{2}$	$\frac{4}{2}$
VBC	AB	BD	**CD**	**CD**	DE	CF	EG
2_{nd} iteration	0	0	0	0	**1**	**1**	$\frac{1}{2}$
VBC	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	**CDE**	**CDF**	CDEG

2.4 Experimental Results

In this section, we evaluate the performance of PCL-VP by comparing it with the approach that builds on the traditional minimum dominating set (MDS), which is named as MDS-based VNF protection (MDS-VP) algorithm. The MDS-VP algorithm keeps greedily selecting a single VB node n without considering the offsite backup constraint, until the VBC can cover all other nodes in SN (excluding the nodes in VBC). Then, the MDS-VP approach finds other physical nodes in Nei_{VBC} to backup each node in VBC.

In the simulation, the network size is set in the range of $\{40, 60, 80, 100, 120\}$, and the connectivity (i.e., node degree) of each physical node is randomly set above 3. Each physical node contains two attributes: computational demands and computing capacity (e.g., CPU), which are both randomly generated. Let the VNF computing load for physical nodes is

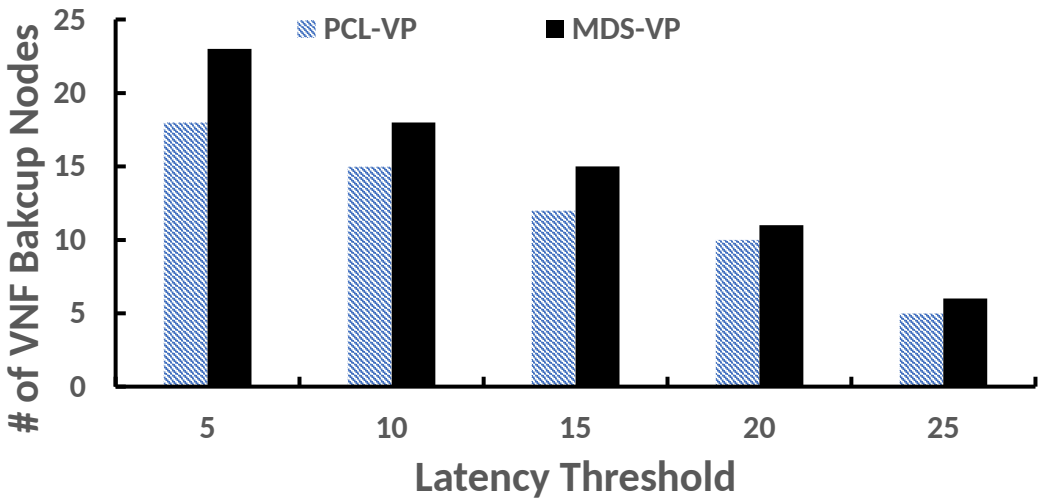


Figure 2.3: $|VBC|$ vs. Latency threshold.

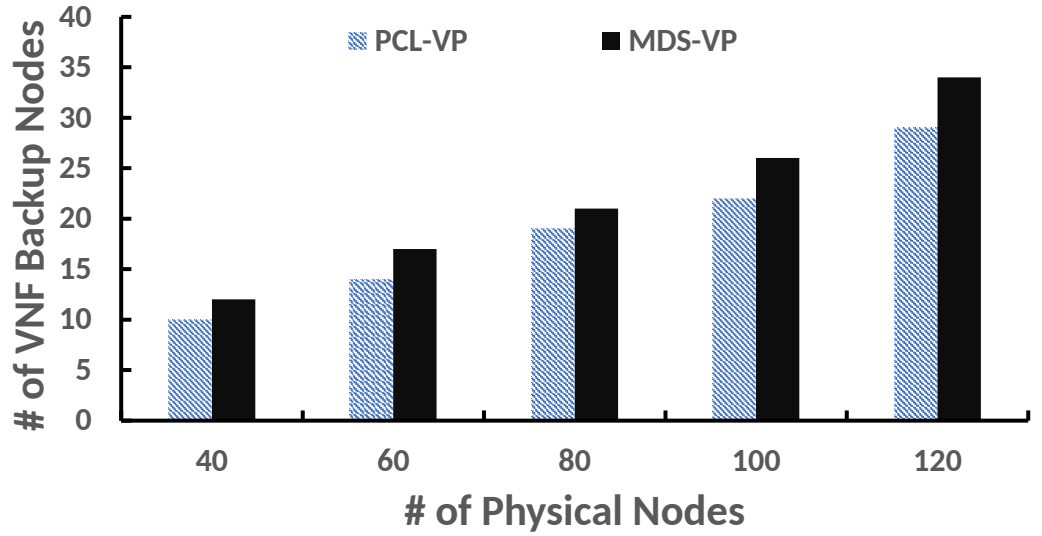


Figure 2.4: $|VBC|$ vs. Physical node.

randomly set in the range of $\{20, 30, 40\}$, while the computing capacity for physical nodes is randomly set as $\{30, 40, 50, 60, 70\}$. There are enough fiber sub-carriers and the link latency is set in the range of [1, 10] with the normal distribution.

Figs. 2.3, 2.4 and 2.5 show the performance of PCL-VP and MDS-VP, where the number

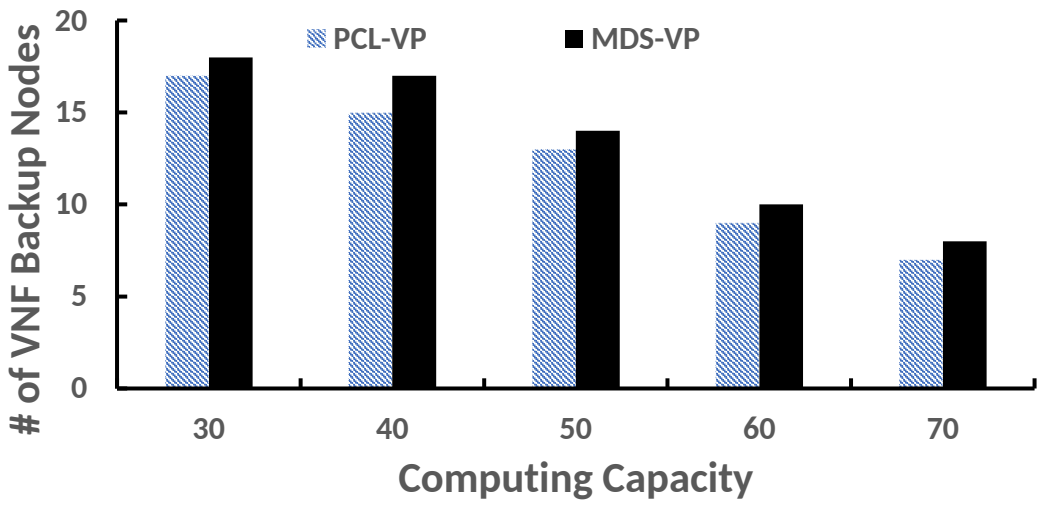


Figure 2.5: $|VBC|$ vs. Computing capacity.

of backup nodes (i.e., the size of VBC set) changes with the latency value, the number of physical nodes, and computing capacity, respectively. The solid black bar in each graph represents the number of VB nodes selected by the MDS-VP approach while the blue dotted bar in each chart represents the number of VB nodes selected by the PCL-VP approach.

Fig. 2.3 demonstrates that when the latency threshold increases, the number of selected VB nodes in VBC decreases for both PCL-VP and MDS-VP, and the difference between two algorithms also decreases. This is because the PCL of each physical node increases when relaxing the latency constraint. Thus, fewer VB nodes are required to protect all the VNF computing load in the whole SN.

When increasing the number of the physical nodes, in Fig. 2.4, the number of selected VB nodes in VBC increases for both PCL-VP and MDS-VP. When the SN has more physical nodes, the difference between the number of selected VB nodes in PCL-VP and MDS-VP becomes larger, which shows the effectiveness of the proposed PCL-VP algorithm.

When increasing the computing capacity, as shown in Fig. 2.5, the more computing capacity owned by the physical nodes, the fewer VB nodes are needed. As one can see that, the proposed PCL-VP algorithm outperforms MDS-VP and the performance difference decreases when increasing the computing capacity of each VB node. This is because the proposed PCL-VP can effectively identify the pairwise protection and the protecting cover list of each VB node increases with the increasing of the computing capacity.

2.5 Summary

In this chapter, we have investigated the VNF failure protection cover (VFPC) problem in EON while considering the latency, computing, and offsite backup constraints. We have formulated a mathematical model for the VFPC problem and proposed an $ln(|N|) + 1$-approximation algorithm, namely, protection cover list-based VNF protection (PCL-VP), where $|N|$ represents the number of physical nodes in the SN. Based on our simulation results, we have shown that the proposed PCL-VP algorithm outperforms the minimum dominating set-based VNF protection (MDS-VP) algorithm. In the future, we will study the survivability of the network service under the shared protection scheme among different SFCs.

CHAPTER 3
SERVICE FUNCTION INSTANCE FAILURE AND PROTECTION IN MFNS

3.1 Motivation

The emerging internet of things and 5G techniques empower many new network services and heterogeneous applications with strict reliability and ultra-low latency requirements [5, 36, 37]. Limited by the computing resource, some applications may offload their computation-intensive tasks to cloud systems that have abundant computing resource but may be far away from the client [38]. Thanks to multi-access edge computing, the servers with enough computing resource can be deployed at the network edge to perform the related processing tasks closer to the client [39]. When network function virtualization (NFV) is applied in multiple-access edge computing, the hardware-based middle-boxes such as firewalls, load balancer, and intrusion-detection system are replaced by service functions (SFs) running on commodity servers [40, 41]. By hosting the instances of SFs on diverse physical nodes in the physical network (PN), NFV can deliver high-performance network services with greater flexibility, scalability, and adaptability at reduced capital expenditures (CAPEX) and operating expense (OPEX) costs when compared to networks built from the traditional middle-boxes [25, 42, 43]. With NFV, a network service request (NSR) from a client consists of a set of SFs and resource demands (e.g., bandwidth, CPU, and storage) [20]. To satisfy an NSR, the service provider creates a service function chain (SFC) by chaining the required SFs into a virtual catenary structure with virtual links, and reserves corresponding networking resource to embed the created SFC [22].

On the one hand, NFV facilitates delivering flexible and cost-efficient services in multi-access edge computing systems. On the other hand, NFV has to confront the challenging reliability and latency requirements in multi-access edge computing applications such as smart factories, stock exchange and autonomous driving [44, 45]. To protect the network from the physical (e.g., hardware appliances) failures that affect the service delivery, the work in [27, 46, 47, 48] proposed several network protection schemes. Even though much research attention has been paid to protecting physical node failures, only little work has focused on the virtual node failures (e.g., the SF failure of an SFC). When SFs are placed on the physical nodes/servers, the virtual node failures can rise due to the misconfigurations or virtual machines crash [12, 13, 14, 15]. Unlike the physical node failures that will change the topology and connectivity of the PN, the virtual node failures may impact the output of the data flows, and break multiple SFCs in the PN [30, 49, 50]. As a result, traditional strategies for the physical node failures cannot be directly applied to efficiently solve such virtual node failures coupling with the low latency requirements from the 5G and beyond 5G communications [29].

In this chapter, we define and investigate a new problem called latency-bounded off-site virtual node protection (LOVNP) in NFV. In LOVNP, all the virtual nodes (i.e., in-service SFs) hosted by a physical node will have the off-site virtual node protection from another physical node that satisfies the latency constraint. We assume that there will be only one physical/virtual node failure at one time, and a failure will be repaired before the next failure happens [51]. Accordingly, we first mathematically formulate the problem of latency-

bounded off-site virtual node protection (LOVNP) with the objective of minimizing the total backup resource consumption in the PN while satisfying the latency requirements. Then, we propose an efficient heuristic algorithm called protection centrality-based pairwise node protection (PC-PNP) to optimize the backup resource consumption, which builds on top of two proposed techniques: the pairwise node protection and the protection centrality techniques. We prove that the proposed PC-PNP algorithm can guarantee a logarithm-approximation bound. Our extensive simulations and analysis show that the proposed algorithm can significantly outperform the methods that are directly extended from the existing work.

The rest of this chapter is organized as follows. We summarize the related work in Section 3.2. In Section 3.3, we introduce the off-site virtual node protection requirement in NFV. We formulate the LOVNP problem with a mathematical model in Section 3.4. In Section 3.5, we analyze the NP-hardness of the proposed LOVNP problem. We propose an efficient algorithm to optimize LOVNP based on two novel techniques in Section 3.6. We evaluate the experimental results in Section 3.7, and conclude our work in Section 3.8.

3.2 Related Work

In this section, we introduce the concepts of SFC embedding in NFV, and discuss the related physical network failures and virtual network failures in NFV.

3.2.1 Service Function Chain Embedding

In NFV, to satisfy a network service request (NSR), the service provider needs to embed the service function chain (SFC) onto the PN. The process of accommodating such an NSR

is referred to as service function chain embedding (SFCE) [52]. When the order of SFs in an SFC is given, the SFCE problem can be regarded as a special case of the virtual network embedding problem [53, 54] or the virtual network function placement problem [55, 56], which has been proved NP-hard. When the order of SFs in an SFC is not specified, the service provider needs to composite the SFC and embed it onto the PN [56, 57].

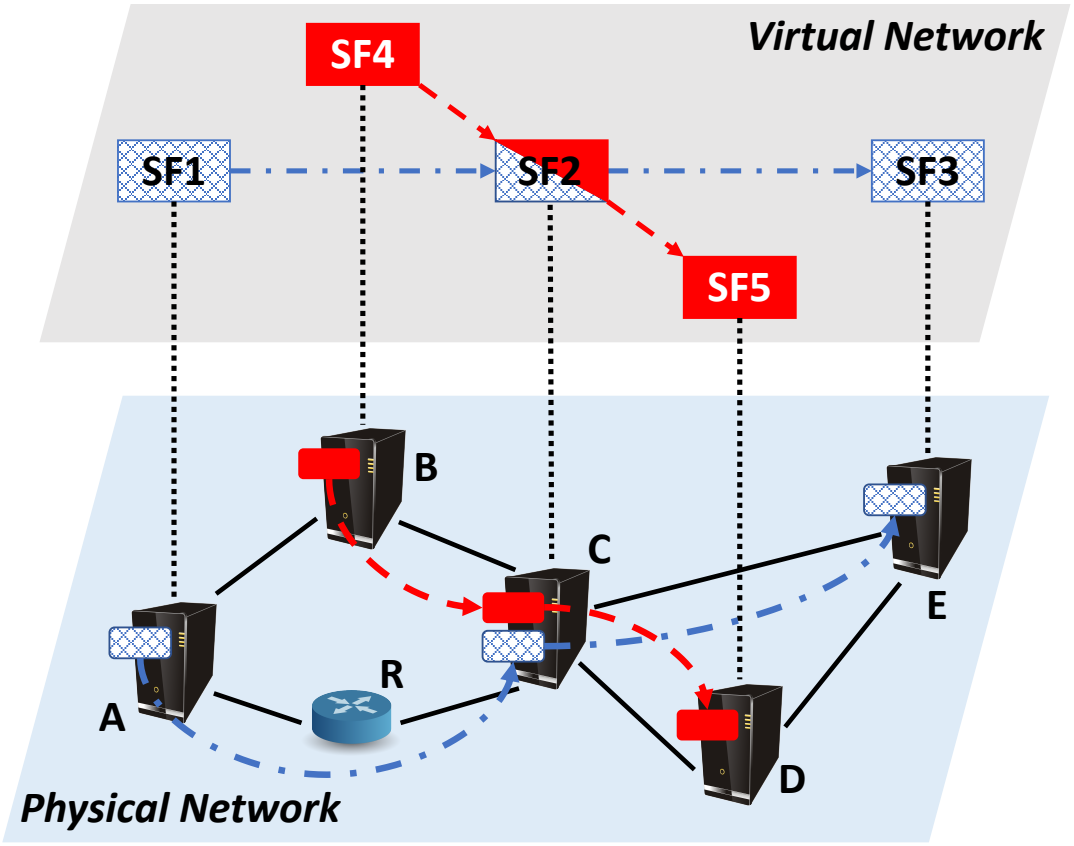


Figure 3.1: Service function chain embedding.

Fig. 3.1 shows an example of the SFCE process, where two given SFCs are embedded onto a PN. In the figure, the given SFCs $SF1 \rightarrow SF2 \rightarrow SF3$ and $SF4 \rightarrow SF2 \rightarrow SF5$ from the clients can be represented as a virtual network. Each node in the virtual network represents a virtual node, and it will be mapped as an SF instance (or in-service SF) on a proper

physical node. For example, the SF instances $SF1$ and $SF3$ are placed onto the physical nodes A and E, respectively. Similarly, the SF instances $SF2$ from two different SFCs can be mapped on the same physical node C. Each link in the virtual network represents a logical link that can be mapped onto a physical path. For example, the virtual link from $SF1 \rightarrow SF2$ in Fig. 3.1 is mapped onto the physical path $A \rightarrow R \rightarrow C$.

3.2.2 Physical Network Failure

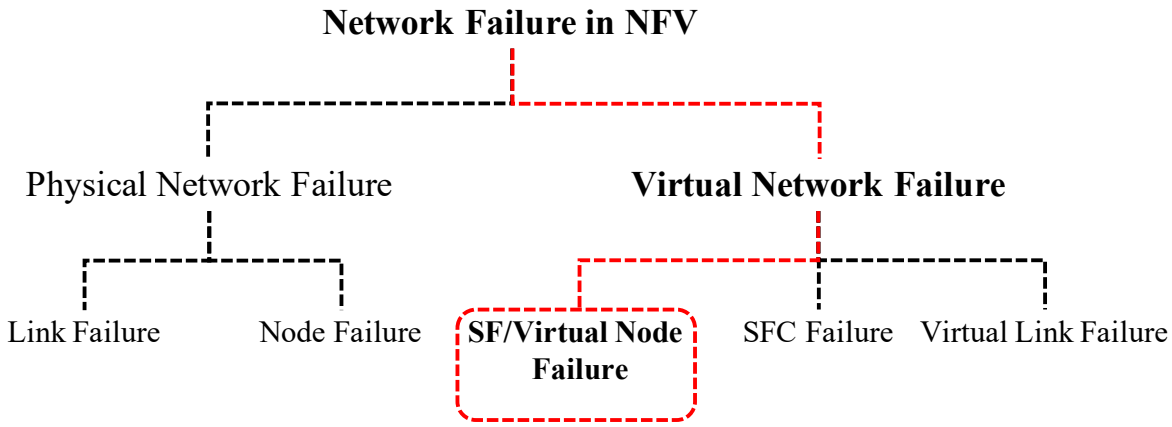


Figure 3.2: Classification of the network failure in NFV.

Fig. 3.2 shows the broad classification of network failures in NFV, which includes (i) physical network failures and (ii) virtual network failures. The physical network failures can be either the physical link failure or the physical node failure in the PN. To recover a physical link failure, one can employ the traditional link protection schemes to reroute all traffic carried by the failed link [58]. For example, the authors in [59] designed backup schemes to protect against random physical link failures, and proposed a probabilistic distribution function to minimize the total backup resource consumption in polynomial time. As the

service provider expects a short recovery time, instead of providing backup paths for all existing flows, the authors in [60] proposed a scheme to find the most efficient backup path against the single link failure based on the group table based rerouting technique.

To protect the physical node failure, the straightforward approach is to identify a set of additional physical nodes to back up the failed nodes. For example, the authors in [46] proposed a backup resource allocation model to jointly consider the backup computing and bandwidth resource allocation against facility node failures. Similarly, in [47], the authors investigated how to conduct backup nodes placement to handle the physical node failures while minimizing the total required backup computing resource. The work from [61] proposed a primary and backup resource allocation model by using the probabilistic protection.

3.2.3 Virtual Network Failure

Recently, there has been increasing interest in the study of virtual network failures in NFV. Virtual network failures can be an SF (or virtual) node failure, an SFC failure or an SF (or virtual) link failure. For example, in Fig. 3.1, the failure of the SF node $SF1$ on the virtual network layer can be due to the crash of the corresponding SF instance. The failure of the virtual link $SF1 \rightarrow SF2$ can happen when the router port of R used for path $A \rightarrow R \rightarrow C$ is damaged. The failure of an SFC can be due to an SF node/link failure or a misconfiguration of the SFC.

For SF failures, the authors in [62] explored the resource-aware backup allocation problem, which aims to minimize the cost of backup nodes while meeting the overall availability demands. The work in [32] investigated the reliable virtual network function service provi-

sioning in the multi-access edge computing system, by provisioning primary and backup SF instances in order to meet the reliability requirements of mobile users. The authors in [63] examined the static and dynamic backup schemes to determine whether the SF backup can be placed on the edge server or the cloud. The work in [31] tried to improve the network reliability against SFC failures by using the SFC parallelism strategy. To recover the virtual link failure, the authors in [64] proposed to provision additional link bandwidth with minimal disruption to the existing traffic.

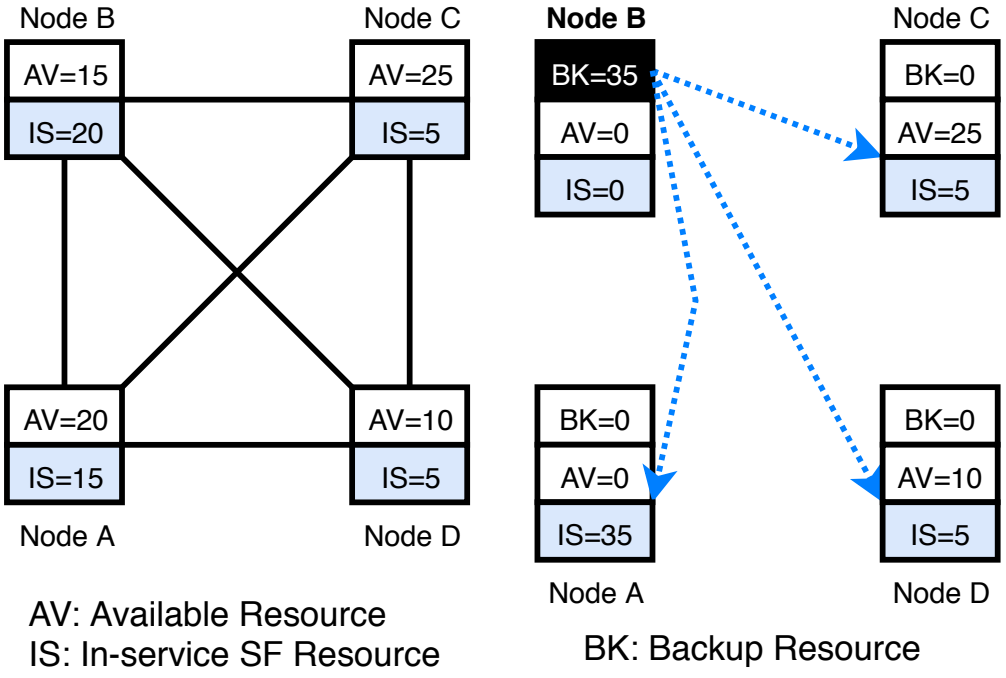
Above all, much work has been done to prevent either physical network failures or virtual network failures. However, no work has jointly taken both physical and virtual layers into consideration in a latency-bounded system. In the following sections, we introduce a new off-site virtual node protection scheme to jointly take the physical node and on-site virtual node protection into account.

3.3 Off-site Virtual Node Protection in NFV

In this section, we assume that a PN is a multiple function mesh network as defined in [17]. In the PN, each physical node can host many in-service SFs. We present the concepts of traditional physical node protection and on-site virtual node protection. Then, we introduce a new NFV protection scheme called off-site virtual node protection (OVNP).

3.3.1 Physical Node Protection

The traditional approach of preventing physical node failures is to find a set of (off-site) backup nodes that are exclusively used for protecting other physical nodes (e.g., [65]).



(a) Physical network.

(b) Physical node protection requires 35 CPU of backup resource.

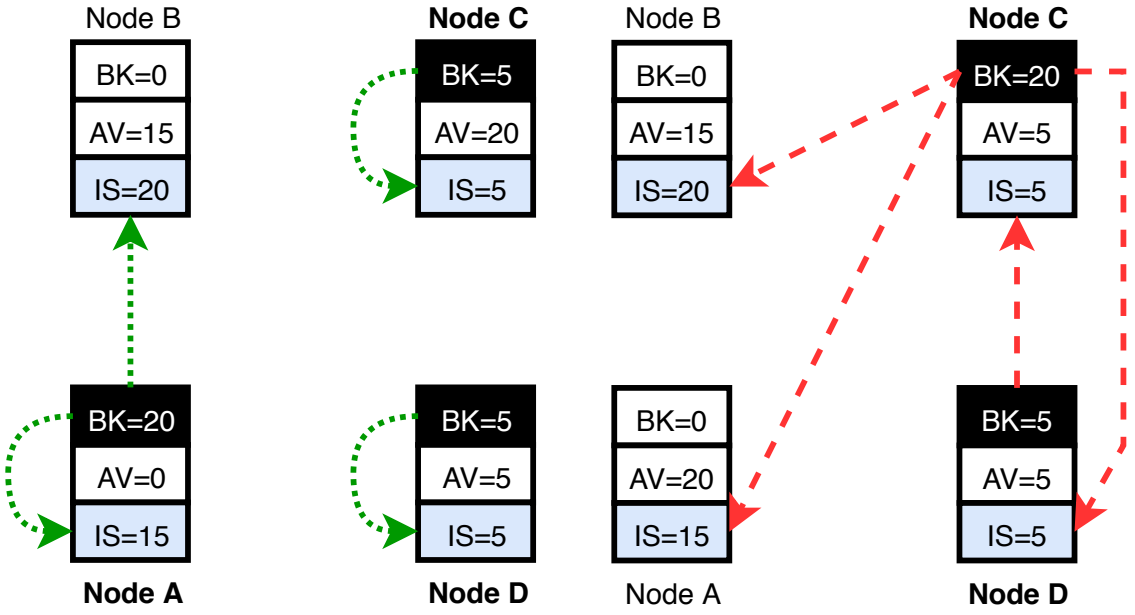
Figure 3.3: Physical network and physical node protection.

Fig. 3.3a represents a small PN that contains four physical nodes, each of which can be an edge server and can host multiple in-service SFs. In specific, for physical node A in Fig. 3.3a, the resource used to run the in-service SFs is 15 CPU, denoted by $IS = 15$ as shown in the blue square. The available resource is 20 CPU (i.e., $AV = 20$) as shown in the white square. Similarly, nodes C and D host in-service (i.e., running) SFs that consume the resource of 5 CPU. To provide protection at only the level of the physical node for a physical node, one would have to find another (i.e., off-site) physical node that has enough resource to backup the entire physical node. For example, in order to use node B protecting other physical nodes in Fig. 3.3a, all SF instances on node B may have to be relocated to node A. Then node B can act as the off-site backup node to provide share protection for node A, C, D as

shown in Fig. 3.3b. The black square represents the allocated backup resource, and the blue dotted arrow shows the protective relationship between the backup node and the protected physical node. Note that, in such a traditional physical node protection scheme, any single SF failure may lead to the entire physical node migrating to the backup node. Hence, for a large-scale network, the service provider may have to set a significant number of backup nodes to ensure network reliability under the latency constraints [51]. In this example, when there are only four physical nodes and assume that all the one-hop links satisfy the latency constraint, it requires one dedicated backup node (i.e., B) and the required backup resource is 35 CPU (the max capacity among A, C, D).

3.3.2 On-site Virtual Node Protection

In NFV, the virtual node (i.e., SF instance) failures are unpredictable, which can be attributed to the misconfiguration, virtual machines crash, or SFs malfunction. Most of the existing studies related to virtual node protection focus on the efficient placement of the backup virtual node [32, 50, 62]. Fig. 3.4a shows an example of a virtual node protection scheme with on-site backup, where all nodes have the same amount of capacity and in-service SFs to be protected as the ones in Fig. 3.3a. The on-site protection scheme prefers using the available resource to backup the in-service SFs within the same physical node. For example, node C in Fig. 3.4a can allocate its own resource ($BK = 5$) to protect the in-service SFs at node C that requires $IS = 5$ CPU of the resource. Similarly, node D can protect the in-service SFs at node D, while node A can protect the in-service SFs at node A. However, node B does not have enough resource available to backup the in-service SFs within itself.



(a) On-site virtual node protection requires 30 CPU of backup resource.

(b) Off-site virtual node protection requires 25 CPU of backup resource.

Figure 3.4: On-site virtual node protection and off-site virtual node protection.

In this case, one would have to find the available resource from another node (e.g., A) to backup the in-service SFs at node B. For example, the virtual node failures on node B can use the backup resource ($BK = 20$) at node A to provide protection for the in-service SFs at node B ($IS = 20$). As a result, the on-site virtual node protection scheme in Fig. 3.4a requires $BK = 20 + 0 + 5 + 5 = 30$ CPU of the backup resource.

3.3.3 Off-site Virtual Node Protection

Table 3.1 shows the primary pros and cons of the aforementioned physical node protection and on-site virtual node protection. One of the major pros of physical node protection is its off-site property. One of the major pros of on-site virtual node protection is its flexibility in

Table 3.1: Pros & cons of physical node, and on-site virtual node protection.

	Pros	Cons
Physical Node Protection	Protect multiple virtual nodes from the same physical node.	May require dedicated physical node resource for protection.
	Guarantee to recover any single physical or virtual node failure.	Reroute the traffic and may increase the latency consumption.
	Provide off-site physical node protection.	Re-instantiate the entire physical node at the backup node when a single virtual node fails.
On-site Virtual Node Protection	Minimize traffic reroute with on-site protection.	Cannot guarantee to protect the physical node failure.
	Save latency with on-site protection.	May require more resource due to less off-site sharing protection.

recovering the individual in-service SF with reduced latency and traffic rerouting. Accordingly, by taking advantage of both physical and on-site virtual node protection, we propose a new shared protection scheme in NFV, called off-site virtual node protection (OVNP). In OVNP, each in-service SF (or virtual node) requires an off-site protection from another physical node. Fig. 3.4b shows an example of the OVNP scheme, where nodes C, D are selected as the backup nodes to protect all in-service SFs in the PN. In this example, the backup resource ($BK = 20$) at node C can provide sharing protection for the in-service SFs at node A ($IS = 15$), node B ($IS = 20$) and node D ($IS = 5$), but cannot provide off-site protection to the in-service SFs at its own (i.e., node C). Hence, to help recover the physical

and virtual node failures on node C, node D ($BK = 5$) would be selected to protect the in-service SFs at node C ($IS = 5$). As a result, the off-site virtual node protection scheme in Fig. 3.4b requires $BK = 20 + 5 = 25$ CPU of the backup resource. The red dashed arrow represents the protective relationship between the backup node and the protected in-service SFs.

In this chapter, we focus on the node failure protection as it is important and needed to increase the network reliability [1]. In order to reduce the potential disruption from the failures, the off-site virtual node protection scheme should also take the latency requirement into account. Accordingly, in the following sections, we propose and investigate a novel problem, called latency-bounded off-site virtual node protection (LOVNP).

3.4 Latency-bounded Off-site Virtual Node Protection (LOVNP)

In this section, we formulate the problem of latency-bounded off-site virtual node protection (LOVNP) in NFV. When a PN and the in-service SFs at each physical node are given, the optimization goal of LOVNP is to minimize the total backup resource consumption for backup nodes.

3.4.1 Physical Network Model

Given a physical network (PN), we can generate an undirected weighted graph $G = (N, L, F)$, where N is the set of physical nodes. Each physical node $n \in N$ can represent an edge server or a data center with a certain amount of available computing resource C_n. The

[1] Link/path protection has been widely studied and is beyond the scope of this work.

set L is the set of auxiliary links. Each link $l_{m,n} \in L$ represents a shortest path with latency $\mathbb{L}_{l_{m,n}}$, where m and n ($\forall m, n \in N$) are its endpoints. We assume that each link has enough bandwidth for supporting data transmission and the protection process. The set F is the set of in-service SFs. For each virtual (or SF) node $f_i \in F$, it requires a certain amount of computing resource D_n.

3.4.2 Problem Formulation

To recover from the physical or virtual node failures on node n, OVNP requires another off-site node $m \in N$ to be the backup node, where $m \neq n$. In addition, to efficiently protect the latency-sensitive applications, the latency between the physical node n and the backup node m must be within the QoS constraint (or service latency boundary).

We define the latency-bounded off-site virtual node protection (LOVNP) problem as: given a set of physical nodes with the available computing resource, and a set of embedded virtual nodes with the required computing resource, how to minimize the total backup resource consumption to protect the PN from virtual node failures while satisfying (i) the QoS latency constraint, (ii) the SF computing constraint, and (iii) the off-site virtual node protection constraint.

Let $cost_m$ represent the cost of the backup node m to protect the in-service SFs on other nodes. Accordingly, the objective function of LOVNP is to minimize the total cost of backup nodes in the PN, which can be written as Eq. (3.1). Let the binary variable $\gamma_m^n = 1$ if SF instances at node n are protected by backup node m, 0 otherwise. Table 3.2 lists the necessary notations.

Table 3.2: Notations.

Notation	Meaning
N	Set of physical nodes.
L	Set of links.
F	Set of required in-service SFs.
n	Physical node $n \in N$.
m	Backup node $m \in N$.
C_n	Available computing resource on node n.
$l_{m,n}$	Link between node m and n.
$\mathbb{L}_{lm,n}$	Latency on link $l_{m,n}$.
f_i	In-service SF $f_i \in F$.
D_n	Needed computing resource for f_i at n.
θ	Latency threshold.
γ_m^n	$= 1$ backup node m can protect node n; 0 otherwise.
$cost_m$	Cost (e.g., backup resource consumption) by m.

QoS latency constraint: As the network delay is relatively dynamic from time to time, a latency threshold θ is set to limit the propagation delay from node n to its backup node m.

SF computing constraint: When there is a physical/virtual node failure occurring at node n, the backup node m must have enough computing resource to recover the in-service SFs on node n.

Off-site virtual node protection constraint: In OVNP, a physical node n with in-service SFs is under off-site virtual node protection if and only if all in-service SFs on node n are protected by another node m ($m \neq n$).

$$\min \sum_{\forall m \in N} cost_m \tag{3.1}$$

Subject to:

$$\gamma_m^n = \begin{cases} 1, & \text{if SF instances at node } n \text{ are} \\ & \text{protected by backup node } m. \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

$$\sum_{\forall m \in N,\ m \neq n} \gamma_m^n = 1, \quad \forall n \in N \tag{3.3}$$

$$\gamma_m^n * \mathbb{L}_{l_{m,n}} \leq \theta, \quad \forall l_{m,n} \in L \tag{3.4}$$

$$\gamma_m^n * D_n \leq C_m, \quad \forall m, n \in N \tag{3.5}$$

$$cost_m = \max_{m,n \in N} D_n * \gamma_m^n, \quad \forall m, n \in N \tag{3.6}$$

Eq. (3.2) determines whether SF instances at node n are protected by the backup node m when there is a direct connection between node n and m. Eq. (3.3) specifies that when the virtual node failure happens, there is another physical node (i.e., the backup node m) to backup the in-service SFs on this node. In Eq. (3.4), θ denotes the latency threshold to ensure the backup node m is not far away from node n. Eq. (3.5) ensures that each backup node m must contain enough computing resource for the in-service SFs that hosted by node n. While the backup node providing share protection, Eq. (3.6) denotes that the cost of the

backup node m is the maximum needed computing resource for the in-service SFs on the protected physical nodes.

3.5 NP-hardness Analysis of LOVNP

In this section, we analyze the NP-hardness of the proposed LOVNP problem in a complete graph and non-complete graph.

3.5.1 OVNP in Complete Graph

When a backup node provides sharing protection to the in-service SFs, the QoS latency constraint limits the feasible protection range for the backup node. To identify the potential backup nodes in the PN, a given undirected graph G can be transferred into graph G' by removing the links that violate the QoS latency constraint.

When the topology of the new transferred graph G' is a complete graph, and each physical node has unlimited computing resource, the optimal number of the backup nodes to protect all in-service SFs among the PN is two (say nodes m and n). node m will protect all in-service SFs other than the ones on its own, and node n will protect the in-service SFs at node m to achieve off-site virtual node protection. These two physical nodes are in fact a pair of backup nodes that need the least computing resource to provide off-site sharing protection for all in-service SFs in the PN. Since there are at most $\mathcal{O}(|N^2|)$ pair of nodes in the PN, Theorem 1 holds.

Theorem 3.5.1. *When G' is a complete graph, and each physical node has unlimited computing resource, the LOVNP problem can be optimized within a polynomial-time.*

Lemma 3.5.2. *When G' is a star or wheel, and each physical node has unlimited computing resource, the LOVNP problem can also be solved in a polynomial-time.*

When G' is a complete graph, each physical node has limited computing resource, and there exists a feasible solution for OVNP, we can also find two backup nodes to protect the PN. We first select the node with the maximum available resource, say node m, to provide sharing protection for all the in-service SFs except its own. Then, another node can be identified to backup node m. Hence, when G' is a complete graph, and there exists a feasible solution for OVNP, the optimal number of the backup nodes to protect all in-service SFs in the PN can be solved in a polynomial time.

Theorem 3.5.3. *If there is a solution for OVNP in a complete graph, then there must exist two backup nodes to optimize the LOVNP problem.*

3.5.2 OVNP in Non-complete Graph

When G' is a non-complete graph by removing invalid links, optimizing the LOVNP problem is NP-hard.

Theorem 3.5.4. *When G' is a non-complete graph and excludes a star or wheel graph, the LOVNP problem is NP-hard.*

Proof. When G' is a non-complete graph, we prove the NP-hardness of the proposed LOVNP problem via the reduction from the minimum weighted dominating set (MDS) problem [66]. Given a PN, when relaxing the off-site virtual node protection constraint, finding a set of physical nodes that can provide sharing protection on all virtual nodes in the PN with the

least computing resource is, in fact, equivalent to the MDS problem. Since the MDS problem is proved to be NP-hard, the LOVNP problem is NP-hard as well. □

Therefore, to optimize LOVNP problem in a multiple functions mesh network, we propose the protection centrality-based pairwise node protection (PC-PNP) algorithm with logarithm-approximation performance in the next section.

3.6 Protection Centrality-based Pairwise Node Protection (PC-PNP)

In this section, we propose the protection centrality-based pairwise node protection (PC-PNP) algorithm to optimize the LOVNP problem. We first introduce the pairwise node protection (PNP) technique that satisfies the off-site virtual node protection constraint, and then, we propose the protection centrality (PC) technique to efficiently identify the physical nodes used for protection.

3.6.1 Pairwise Node Protection (PNP)

Under the constraint of Eq. (3.3) in Section 3.4, each selected backup node in the PN will be protected by another backup node. However, greedily applying the traditional backup nodes selecting policies (e.g., largest degree first, least backup computing resource first) to protect the in-service SFs in the PN may end up with more resource consumption for the sake of the off-site virtual node protection constraint. To guarantee that the selected backup nodes always satisfy the off-site virtual node protection constraint, we propose a novel Pairwise Node Protection (PNP) technique. Here, in order to facilitate the presentation of the proposed PNP technique, we define three sets: backup node set BS, the protected node

set PS, and the unprotected node set UPS. The set of BS includes the identified backup nodes that already allocate backup resource to provide sharing protection for the in-service SFs in PS. The set of PS consists of physical nodes that are protected by the backup nodes in BS while the physical nodes in UPS have not identified the backup nodes/resource yet. In other words, the backup nodes in BS will provide sharing protection for nodes in PS, while the nodes in UPS need to identify backup nodes/resource from N. During the process of identifying backup nodes/resource, the relations of $PS \cup UPS = N$, $PS \cap UPS = \emptyset$ and $BS \subseteq PS$ are always true.

In PNP, the system starts with $BS = \emptyset, PS = \emptyset, UPS = N$. The goal of PNP is to move each node $n \in UPS$ to PS after identifying the proper backup node/resource (for n) in BS. To efficiently identify the backup node and allocate the backup resource, we propose to identify a pair of physical nodes (i, j), where $i \in N - BS, j \in UPS$ in each iteration, such that all the in-service SFs in these two nodes will satisfy the off-site virtual node protection constraint in Eq. (3.3). In other words, at the end of the iteration, all the in-service SFs at node i and j should be under protection from BS. Specifically, two scenarios will arise when identifying (i, j), where $i \in N - BS$ and $j \in UPS$: (a) $i \in UPS, j \in UPS$ and (b) $i \in PS - BS, j \in UPS$. For the former scenario, no nodes in PS can offer protection for these two nodes. Hence, these two nodes from UPS will have to mutually protect each other and move from UPS into PS. In addition, these two nodes will allocate proper backup resource and are added into BS. For the latter scenario, we will use node $i \in PS - BS$ to protect node $j \in UPS$. As a result, node j will be moved from UPS into PS while node i

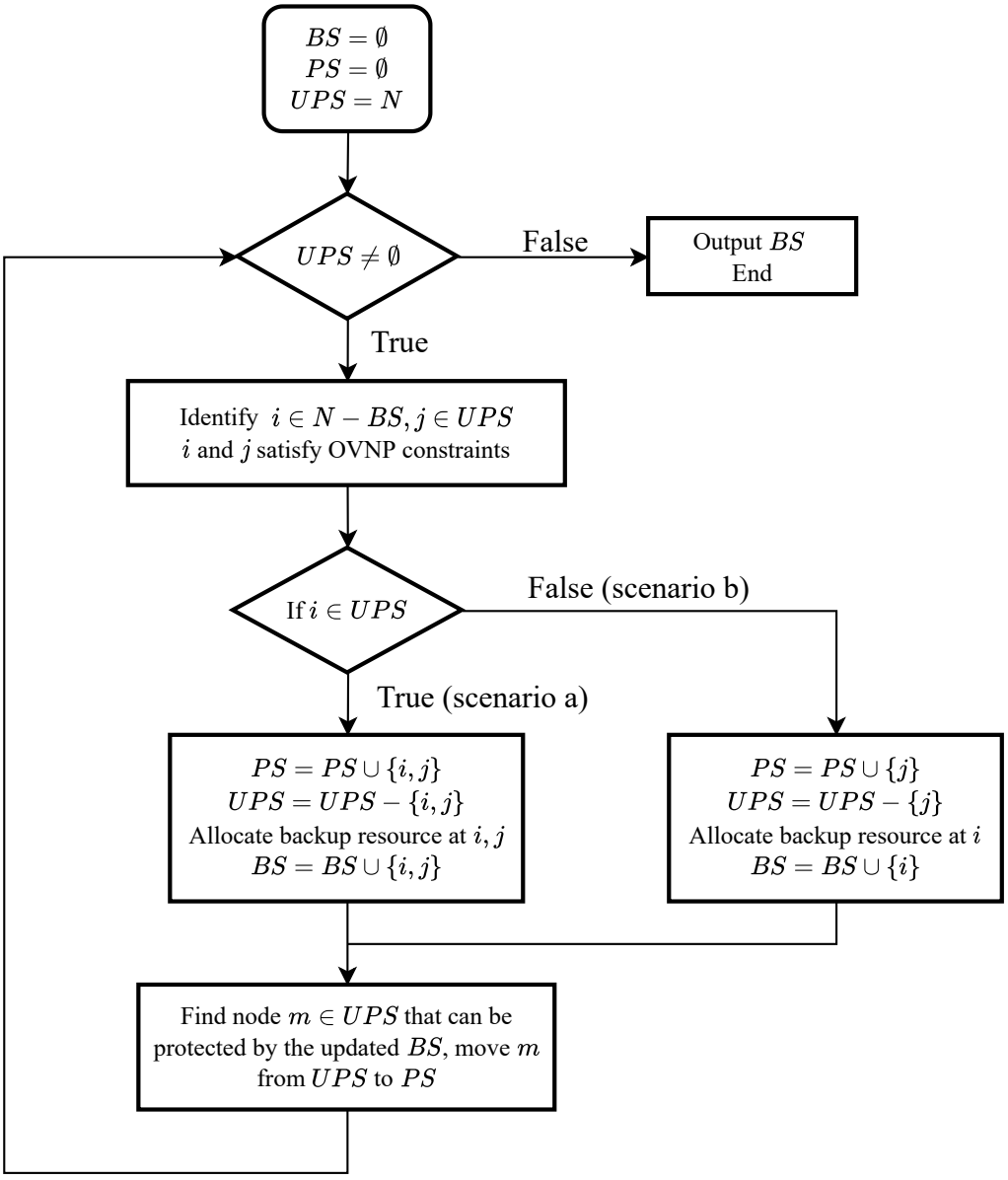
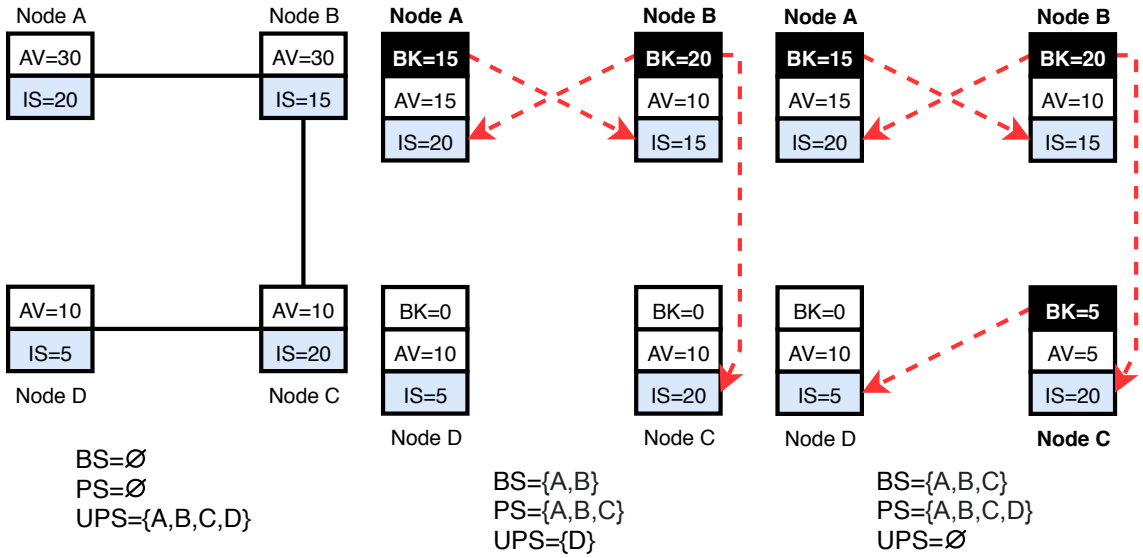


Figure 3.5: Flowchart for PNP.

will allocate proper backup resource (for j) and is added into BS.

Fig. 3.5 shows the flowchart for the proposed PNP technique. Note that within each iteration, whenever more nodes are added into BS, PNP will move node m from UPS to PS if node m can be protected by the updated BS automatically (i.e., without consuming

additional backup resource).



(a) Physical Network. (b) 1^{st} iteration. (c) 2^{nd} iteration.

Figure 3.6: Examples of the PN and PNP selection.

Fig. 3.6 shows an example of how the PNP technique is applied. Fig. 3.6a is a PN with four physical nodes, while Fig. 3.6b and Fig. 3.6c are the PNP technique results from iterations 1 and 2, respectively. In the 1^{st} iteration as shown in Fig. 3.6b, PNP will identify node pair (A, B) from UPS that can mutually protect each other (i.e., scenario a). Nodes A and B will be moved from UPS into PS and added into BS after backup resource allocation. At the same time, node C will be moved into PS since the updated $BS = \{A, B\}$ can offer protection for C automatically. For the next iteration, node pair (C, D) will be identified (i.e., scenario b), where C needs to allocate backup resource to protect D. At the end of PNP, all nodes are in PS, and BS will be the backup node set. Note that, the QoS latency requirement should also be satisfied (i.e., Eq. (3.4) in Section 3.4). As a result, Eq. (3.7) must hold for any node pairs identified by the PNP technique.

$$\mathbb{L}_{i,j} \leq \theta \tag{3.7}$$

In addition, for scenario (a), when both backup nodes are selected from UPS, according to the off-site virtual node protection constraint, they should provide enough available computing resource to protect each other as Eqs. (3.8) and (3.9). For scenario (b), since node i has already been protected by the node in BS, Eq. (3.9) needs to ensure that node i can offer protection for node j from UPS.

$$c_j - D_i \geq 0 \tag{3.8}$$

$$c_i - D_j \geq 0 \tag{3.9}$$

Clearly, the above PNP technique will guarantee that the off-site virtual node protection constraint can always be satisfied. However, to optimize the LOVNP problem, we need to investigate how to properly identify the node pair in PNP, such that the overall backup computing resource consumption can be minimized. Next, we propose the protection-centrality (PC) technique to identify the potential backup pairs that can efficiently provide off-site sharing protection.

3.6.2 Protection Centrality (PC)

As illustrated above, the proposed PNP technique finds a pair of nodes while not violating the off-site virtual node protection constraint. However, randomly selecting a pair of nodes

may not optimize the backup resource consumption. To minimize the backup computing resource consumption, we propose the protection centrality (PC) technique to facilitate PNP in identifying node pairs for efficient network protection.

Intuitively, the backup computing resource used to protect the entire network is closely related to the number of selected backup nodes. As the network topology is fixed, to efficiently protect the entire network with the least computing resource, the pair of backup nodes should (i) protect as many nodes from UPS as possible, and (ii) cost as less computing resource as possible. Accordingly, we define a factor, namely, protection centrality (PC), where the PC value of each pair of nodes is calculated as Eq. (3.10).

$$PC_{i,j} = \max \frac{|(P_i \cup P_j) - (P_i \cup P_j \cap PS)|}{\mathbb{C}_\xi} \tag{3.10}$$

In Eq. (3.10), we use P_i and P_j to represent the set of nodes that can be protected by the potential backup nodes i and j, respectively. Note that, $P_i \cup P_j \cap PS$ represents the set of physical nodes that have been protected by the backup nodes in BS. The numerator of the fraction in fact represents the number of physical nodes in UPS that will be protected by i and j. For the denominator, $\mathbb{C}_\xi$ represents the needed computing resource for protecting these nodes. The PC value of a pair of nodes (i, j) indicates how many physical nodes in UPS will be protected per unit of the computing resource. The higher the PC value, the more effective the two potential backup nodes (i, j) will be in providing protection for the PN.

3.6.3 Protection Centrality-based Pairwise Node Protection (PC-PNP)

We combine the proposed PNP and PC techniques to propose the protection centrality-based pairwise node protection (PC-PNP) algorithm as Alg. 4. To begin with, PC-PNP initializes the backup set BS and the protected node set PS as empty. Then, PC-PNP repeats the following operations until $UPS == \emptyset$ (Lines 4-19): (i) applies the PNP technique to find all potential back nodes and save them as feasible backup pairs set (FS) (Line 5); (ii) updates the PC value for each pair of potential backup nodes in FS according to Eq. (3.10) (Line 6); and (iii) PC-PNP selects the pair of potential backup nodes from FS with the highest PC value, and adds them to BS (Lines 7 - 18). At last, the PC-PNP algorithm returns BS and the corresponding backup resource consumption.

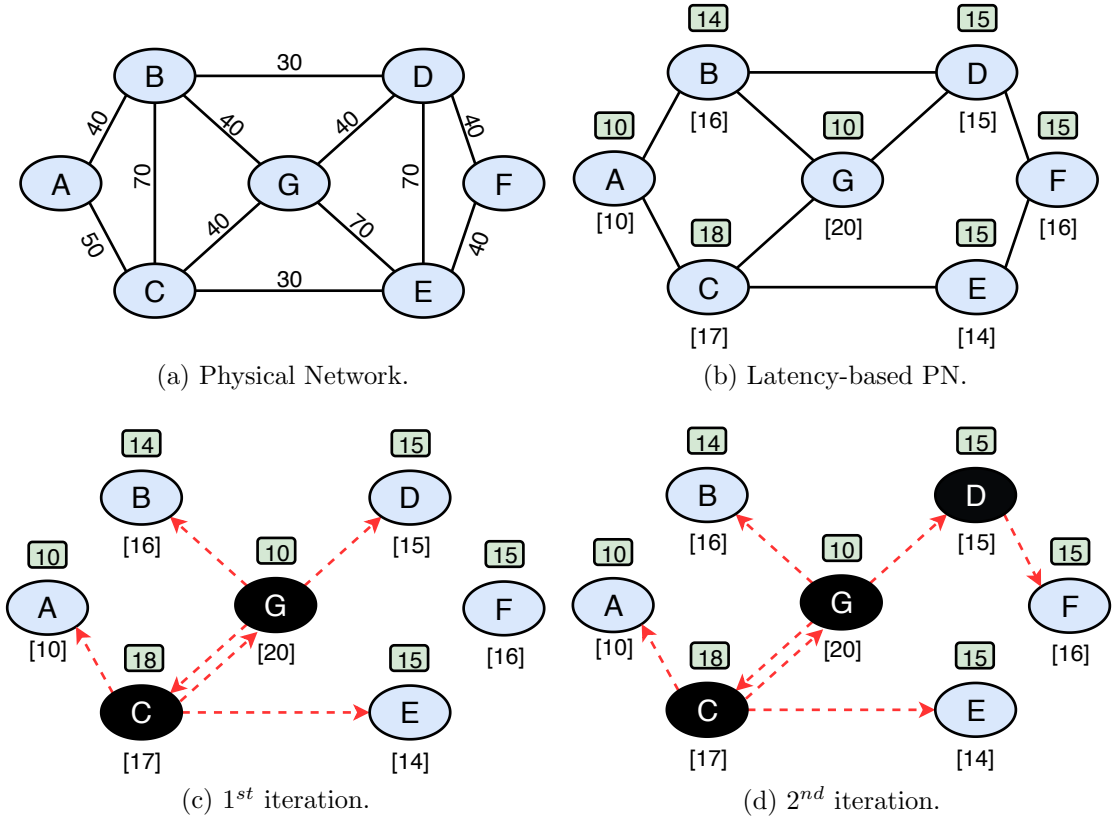
Next, we illustrate how the proposed PC-PNP algorithm works with the examples in Fig. 3.7. Fig. 3.7a shows a given PN with seven physical nodes, and the number next to each link represents its latency. We assume that the latency threshold is $\theta = 50\ \mu s$. The original PN will be transferred into the latency-based PN as shown in Fig. 3.7b. In Fig. 3.7b, the number in the square brackets under each node represents the available computing resource, while the number in the green box above each node represents the needed computing resource for running SFs. In Fig. 3.7c and 3.7d, the red dashed arrow represents the protective relationship between the backup node and the protected physical node, where the backup node is in dark while the physical nodes under protection are in blue. In the 1^{st} iteration of Table 3.3, the PC-PNP algorithm will pick (C, G) as the node pair. This is because the PC value of C and G is the highest. As a result, A, B, C, D, E, G are protected by (C, G)

Algorithm 4 Protection Centrality-based Pairwise Node Protection (PC-PNP) Algorithm

Input: G';
Output: BS, $cost$;
Initialize $BS = \emptyset, PS = \emptyset$, and $UPS = N$;
while $UPS \neq \emptyset$ **do**
    Find all pairs of potential backup nodes by applying the PNP technique, and save them as FS set;
    Update the PC values for all pairs of potential backup nodes in FS according to Eq. (3.10);
    Select the pair of potential backup nodes i, j with the highest PC value from FS;
    if $(i, j \in UPS)$ **then**
        Allocate backup resource at i, j;
        Add i, j into BS;
        Move i, j from UPS into PS;
    else
        Allocate backup resource at i;
        Add i into BS;
        Move j from UPS into PS;
    end if
    Find nodes in UPS that can be protected by the updated BS and move them into PS;
    $cost = cost + \mathbb{C}_{\xi}$
end while
return BS, $cost$;

with 33 CPU of the backup resource. PC-PNP adds both nodes C, G into BS, and updates $BS = \{C, G\}$, $PS = \{A, B, C, D, E, G\}$ and $cost = 33$. In the 2^{nd} iteration, since the unprotected node set $UPS = \{F\}$, node pair (D, F) with the highest PC value is identified, where D needs to allocate backup resource to protect F. Thus, PC-PNP adds node D into BS, and updates $BS = \{C, G, D\}$, $PS = \{A, B, C, D, E, G, F\}$ and $cost = 33 + 15 = 48$. At last, PC-PNP returns the selected backup nodes (i.e., C, G, D) while the total backup resource consumption is 48 CPU.

As one can see that the calculation of the PC value depends on the number of nodes



(a) Physical Network. (b) Latency-based PN.

(c) 1^{st} iteration. (d) 2^{nd} iteration.

Figure 3.7: Examples of PC-PNP algorithm.

Table 3.3: PC values for backup pairs in FS in each iteration.

Backup Pairs	B,D	B,G	**C,G**	D,F	D,G
1_{st} iteration	$\frac{5}{30}$	$\frac{5}{33}$	$\frac{\mathbf{6}}{\mathbf{33}}$	$\frac{5}{30}$	$\frac{5}{33}$
Backup Pairs	B,D	B,G	C,G	**D,F**	D,G
2_{nd} iteration	0	0	0	$\frac{\mathbf{1}}{\mathbf{15}}$	0

that can be protected by the node pair (i, j) and the computing resource required by these nodes. A node pair (i, j) protects at most all physical nodes in the PN (i.e., the number of protected nodes is $|N|$). It takes at most $\frac{|N|(|N|-1)}{2}$ times to traverse all potential node pairs and calculate the PC value of each pair, which is $\mathcal{O}(|N|^2)$. After that, at most $|N|$ iterations are required to find all backup nodes such that all nodes in the PN are protected. Thus, in

Table 3.4: Notations.

Notation	Meaning
Δ	New selected backup node(s) in each iteration.
T	Total iterations of PC-PNP algorithm.
$Cost_i$	Backup resource consumption of PC-PNP in iteration i.
OPT_i	The optimal solution of LOVNP problem in iteration i.
$\|\Gamma_i\|$	Number of protected nodes in iteration i.
$\|U_i\|$	Number of unprotected physical nodes in iteration i.

the worst case, the PC-PNP algorithm has the time complexity as $\mathcal{O}(|N|^3)$.

3.6.4 Boundary Analysis

Next, we prove that PC-PNP can achieve a logarithm-approximation. To facilitate the presentation for the bound proof, the necessary notations are given in Table 3.4.

Theorem 3.6.1. *PC-PNP algorithm can guarantee an $ln(|N|)$-approximation performance, where $|N|$ is the size of the PN.*

Proof. Suppose that the proposed PC-PNP algorithm requires T iterations to select the proper backup nodes to protect the entire PN. We denote the specific iteration as t, and let OPT_t denote the optimal solution of LOVNP problem in the t^{th} iteration. Let Γ_t indicate the set of physical nodes that protected by the PC-PNP algorithm in t^{th} iteration. Let P_Δ represent the set of physical nodes that are protected by backup node(s) Δ in t^{th} iteration. $Cost_t$ represents the backup resource consumption in t^{th} iteration.

The relationship of $|\Gamma_t|$ and $|P_\Delta|$ can be expressed by Eq. (3.11), where $|\Gamma_t|$ and $|P_\Delta|$ represent

the cardinality of Γ_t and P_Δ, respectively.

$$|\Gamma_t| = \bigcup |P_\Delta|,\ \forall t \in [0, T], \Delta \in N \tag{3.11}$$

Let $|U_t|$ represent the number of unprotected physical nodes in the t^{th} iteration. The relationship among $|U_t|$, $|U_{t+1}|$ and $|\Gamma_t|$ can be described as Eq. (3.12).

$$|U_t| = |\Gamma_t| + |U_{t+1}|,\ \forall t \in [0, T] \tag{3.12}$$

$|U_0|$ (the unprotected physical nodes in 0^{th} iteration) equals to the size of the PN as shown in Eq. (3.13).

$$|U_0| = |N| \tag{3.13}$$

Hence, in Eq. (3.14), the fraction on the left is the PC value for Γ_t in t^{th} iteration, while the fraction on the right represents the average cost for each backup node in OPT_t. Since the proposed PC-PNP algorithm guarantees the selected backup node(s) owns the highest PC value, Eq. (3.14) holds.

$$\frac{|\Gamma_t|}{Cost_t} \geq \frac{|U_t|}{OPT_t},\ \forall t \in [0, T] \tag{3.14}$$

Then, one can transfer Eq. (3.14) into Eq. (3.15).

$$Cost_t \leq \frac{|\Gamma_t|}{|U_t|} * OPT_t,\ \forall t \in [0, T] \tag{3.15}$$

Based on Eq. (3.12), we can have Eq. (3.16).

$$\begin{aligned} \frac{|\Gamma_t|}{|U_t|} &\leq \frac{1}{|U_t|} + \frac{1}{|U_t| - 1} + \ldots + \frac{1}{|U_t| - |\Gamma_t| + 1} \\ &\leq \sum_{|U_{t+1}| \leq \delta \leq |U_t|} \frac{1}{\delta},\ \forall t \in [0, T] \end{aligned} \tag{3.16}$$

When accumulating the results of PC-PNP in T iterations, Eq. (3.15) can be transferred into Eq. (3.17).

$$\sum_{t=0}^{T} Cost_t \leq \sum_{t=0}^{T} (\frac{|\Gamma_t|}{|U_t|} * OPT_t),\ \forall t \in [0, T] \tag{3.17}$$

According to Eq. (3.15), Eq. (3.17) can be formatted as Eq. (3.18); in particular, the left side of Eq. (3.17) is the solution from the proposed PC-PNP algorithm.

$$\sum_{t=0}^{T} Cost_t \leq \sum_{|U_{t+1}| \leq \delta \leq |U_t|} \sum_{t=0}^{T} (\frac{1}{\delta} * OPT_t)$$

$$\leq \sum_{|U_T| \leq \delta \leq |U_0|} (\frac{1}{\delta} * OPT) \tag{3.18}$$

According to the property of harmonic series, Eq. (3.18) can be further extended to Eq. (3.19):

$$Cost_{\text{PC-PNP}} = \sum_{t=0}^{T} Cost_t \leq (ln(|N|) + 1) * OPT \tag{3.19}$$

Thus, the PC-PNP algorithm result is bounded by an $ln(|N|)$-approximation, where $|N|$ is the size of the PN. □

3.7 Experimental Results

3.7.1 Simulation Settings

We evaluate the performance of the proposed algorithm through extensive simulations. Similar to the simulation setup in [17, 32, 48], we conduct simulations on large-scale networks that are randomly generated with different node connectivity. Unless otherwise specified, the

physical network is randomly generated with the number of nodes in the range of $[1000, 5000]$. The connectivity (i.e., the number of directly connected neighbors) of each node in the PN is randomly set in the range of $[5, 25]$. Each physical node's available computing resource and the amount of computing resource of each virtual node are randomly set in the range of $[30 - 60]$. For each link, the link bandwidth is randomly set in the range of $[5, 10]$, and the link latency is randomly set in the range of $[1 - 20]$.

3.7.2 Evaluation Metrics

We use the following metrics to evaluate the performance of the proposed algorithms.

Cost rate factor (CRF): We define the cost rate factor (CRF) as $CRF = \frac{cost_{BS}}{|N|}$, where $cost_{BS}$ and $|N|$ are the total cost of BS and the total number of physical nodes in the PN.

Number of backup nodes: The number of backup nodes is the total number of backup nodes to protect the node failures in the PN.

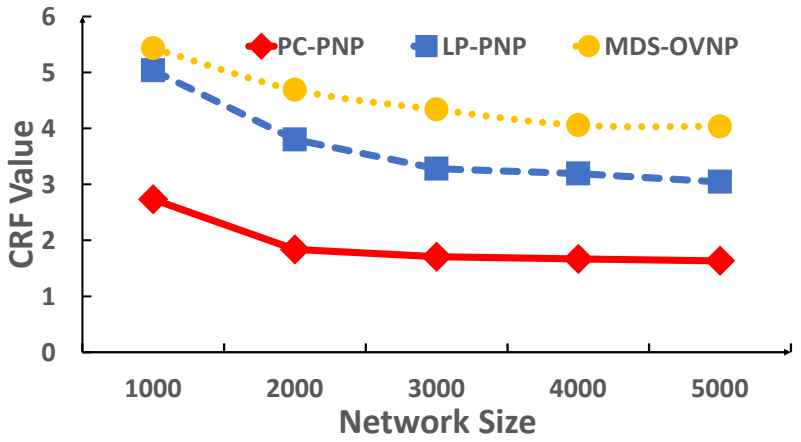
For the evaluation, we compare the performance of the proposed PC-PNP with two greedy algorithms: MDS-based off-site virtual node protection (MDS-OVNP) and largest pair first pairwise node protection (LP-PNP). The basic steps of the MDS-OVNP algorithm are as follows: (i) greedily selects a single backup node i with the largest degree in each iteration, adds i into BS and moves the protected nodes from UPS into PS; (ii) repeats step (i) until $PS = N$; (iii) for each node $i \in BS$, identifies another node $j \in N$ such that node j ($j \neq i$) satisfies the latency threshold and can provide off-site protection for node i, adds node j into BS when necessary; (iv) calculates the total cost of BS.

In LP-PNP, the backup nodes selection takes the number of neighbors and the maximum

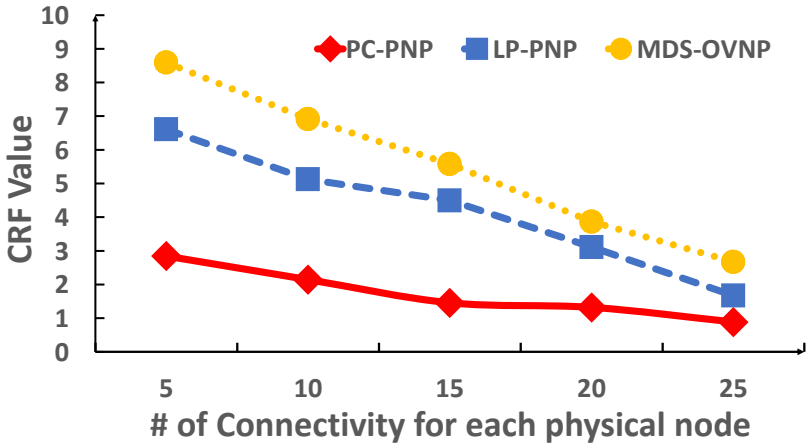
needed computing resource of each pair into account. We define $LP_{i,j} = \frac{|P_i \cup P_j|}{\mathbb{C}_i + \mathbb{C}_j}$, where $|P_i \cup P_j|$ and $\mathbb{C}_i + \mathbb{C}_j$ are the number of protected neighbors and the maximum needed computing resource, respectively. In LP-PNP, the algorithm keeps selecting a pair of nodes with the largest LP value until the network is fully under protection.

3.7.3 Impact of Network Size

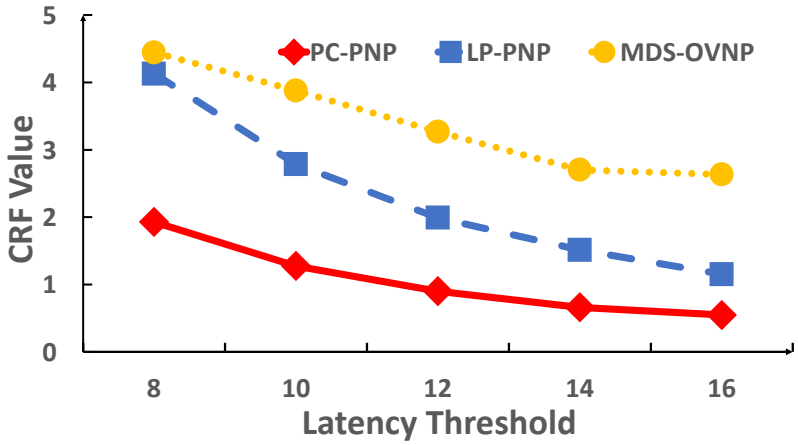
In Fig. 3.8a, we set the latency threshold $\theta = 6$ and change the size of the generated network. The performances of MDS-OVNP, LP-PNP and PC-PNP are denoted by the yellow circle-dotted, blue square-dashed, and red rhombus-solid curves, respectively. As one can see that the CRF values of three algorithms decrease when the network size increases. With the increasing size of the network, more backup node pairs can be selected to protect the network, leading to more sharing opportunities among the selected node pairs, thus decreasing the average CRF values. Note that, the proposed PC-PNP algorithm outperforms the other two schemes by as much as 48.1% and 58.1%. It is because the proposed protection centrality technique tries to maximize the protection sharing by each backup node pair in each iteration. This is further verified by the simulation results in Fig. 3.9a, which shows the number of required backup nodes. In Fig. 3.9a, the red, blue and yellow bars represent the number of backup nodes selected by PC-PNP, LP-PNP, and MDS-OVNP approaches, respectively. The proposed PC-PNP can protect the entire network with less number of backup nodes due to the fact the proposed techniques of pairwise node protection and protection centrality can efficiently identify backup nodes and maximize backup sharing for off-site virtual node protection.



(a) The CRF value vs the network size.

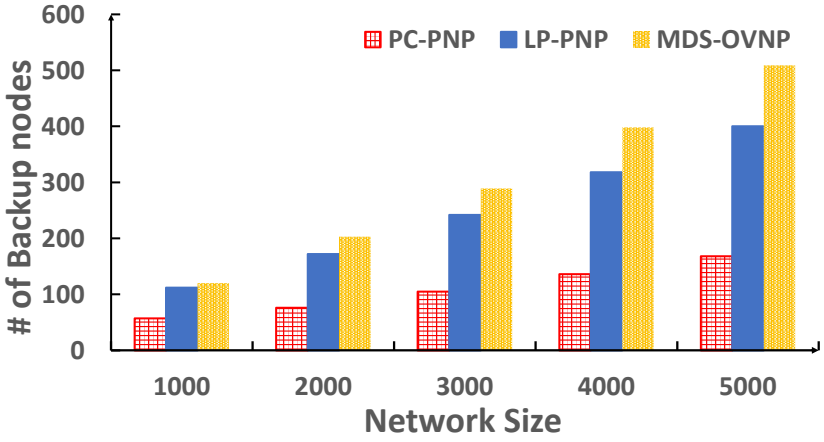


(b) The CRF value vs # of connectivity.

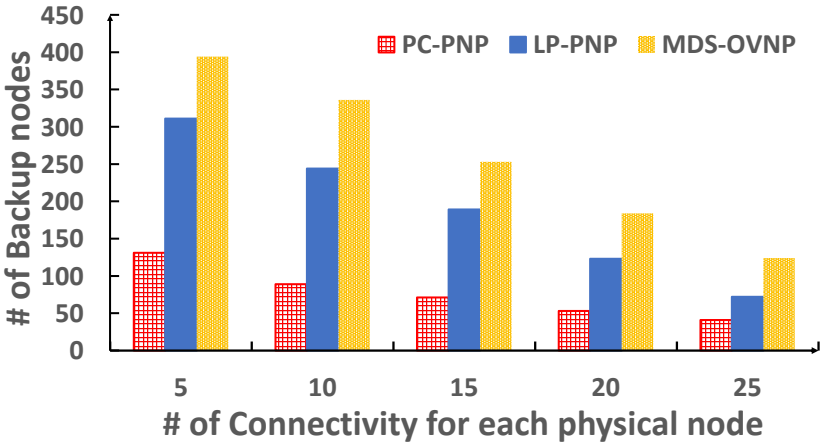


(c) The CRF value vs latency threshold.

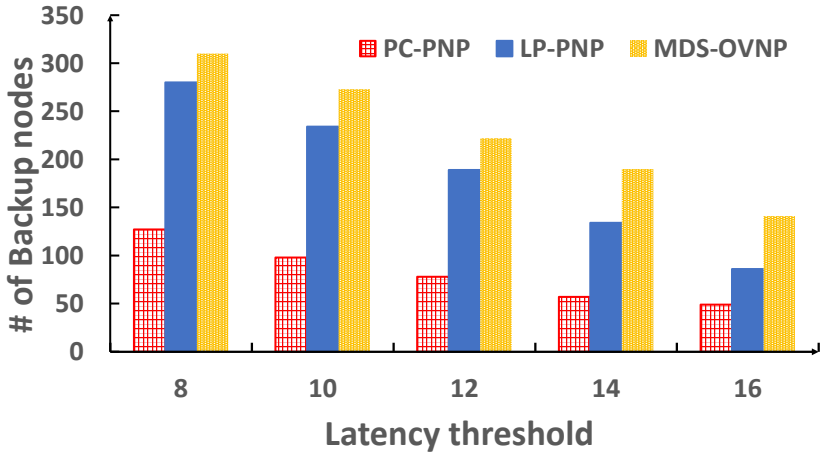
Figure 3.8: Cost rate factor (CRF) values of PC-PNP, LP-PNP and MDS-OVNP.



(a) # of backup nodes vs the network size.



(b) # of backup nodes vs # of connectivity.



(c) # of backup nodes vs latency threshold.

Figure 3.9: Number of required backup nodes of PC-PNP, LP-PNP and MDS-OVNP.

3.7.4 Impact of Connectivity

In Fig. 3.8b and Fig. 3.9b, we generated networks with different node connectivity. As one can see from Fig. 3.8b, when the number of adjacent neighbors (or node connectivity) increases, the CRF values from all three algorithms decrease.

This is because more node connectivity will enable a backup node to provide sharing protection for more virtual neighbor nodes. Similarly, the number of required backup nodes for the three algorithms in Fig. 3.9b decreases with larger node connectivity. In addition, Fig. 3.8b and Fig. 3.9b show that LP-PNP is better than MDS-OVNP by as much as 27.4%. This is due to the fact that the technique of pairwise node protection technique can efficiently identify a pair of virtual nodes that can satisfy the off-site virtual node protection constraint during each iteration. When combining the pairwise node protection technique with the protection centrality technique, the proposed PC-PNP outperforms LP-PNP by as much as 57.4%. This is because the protection centrality technique will effectively select node pairs that maximize the backup sharing.

3.7.5 Impact of Latency Threshold

Fig. 3.8c and Fig. 3.9c show when increasing the value of the latency threshold θ, the CRF values and the total number of backup nodes decrease for all three algorithms. Relaxing the latency threshold will enlarge the set size of potential backup nodes. Hence, more potential backup sharing can lead to a decrease in the number of backup nodes and backup cost for all schemes. Interestingly, in Fig. 3.8c, when θ is smaller than 9, the CRF

values of LP-PNP and MDS-OVNP are relatively close. This is because when the latency threshold is small, the adjacent neighbors of each node are limited and both algorithms have very limited nodes in PS for backup sharing. In other words, both algorithms cannot efficiently identify proper backup sharing while satisfying the off-site virtual node protection constraint. As a result, the CRF value from both algorithms are approaching the one from dedicated protection (i.e., double of the amount of the computing resource required by all in-service SFs). Overall, the proposed PC-PNP algorithm averagely outperforms MDS-OVNP by 62% and outperforms LP-PNP as much as 58% in terms of the CRF values.

3.8 Summary

In this chapter, we have investigated a new latency-bounded off-site virtual node protection (LOVNP) problem, which considers the SF computation, the QoS latency, and the off-site virtual node protection constraints. We have proved the NP-hardness of the LOVNP problem and proposed an efficient logarithm-approximate algorithm, called protection centrality-based pairwise node protection (PC-PNP). The proposed PC-PNP algorithm employs two novel techniques: pairwise node protection and protection centrality. The former ensures node pairs are efficiently identified to meet the demands of the off-site virtual node protection constraint, while the latter tries to maximize the backup sharing among the selected backup nodes. We have also proved that the proposed PC-PNP algorithm is logarithm-approximate. Our extensive simulation and analysis have shown that the proposed PC-PNP algorithm outperforms the algorithms that are extended from the existing

work. In this chapter, we have only considered the cases with a single virtual node failure, and a physical node can support all types of SFs. In future work, we will investigate more practical cases with multiple failures and limited link bandwidth. Similarly, further studies are deserved when each physical node can only support a limited set of SFs.

CHAPTER 4

SERVICE FUNCTION FORWARDER FAILURE AND PROTECTION IN CLOUD NETWORKS

4.1 Motivation

The wide deployment of 5G cellular infrastructures allows ultra-reliable and low-latency communication (URLLC) services being ubiquitously deployed at the network edge [67]. Delivering the URLLC service needs a set of network functions, which are generally implemented in proprietary middleboxes [68]. As the maintenance and management of these dedicated middleboxes are expensive, network function virtualization (NFV) is introduced to implement these functions with software-based modules, called service functions (SFs) [69]. In NFV, a service function request (SFR) is comprised of multiple SFs and related resource demands (e.g., CPU, bandwidth) [21]. To deploy an SFR, the service providers bundles the required SFs into a service function chain (SFC) and embeds it onto a shared physical network (PN) [70]. The process of composing and embedding an SFC is referred to as service function chaining and embedding (SFCE) [43]. The physical forwarding path that hosts the SFC is referred to as a service function path (SFP) [71]. Along an SFP, every SF instance is initiated to run a specific SF and connects to a service function forwarder (SFF) [72].

Recently, due to the significant financial losses caused by software failures, great efforts have been put into investigating the SF protection in failure-sensitive applications, such as real-time stock markets, remote surgery, and software-driven vehicles [73, 74, 75, 76]. Even though many works have proposed protection approaches for SF failures [77, 78, 79], few

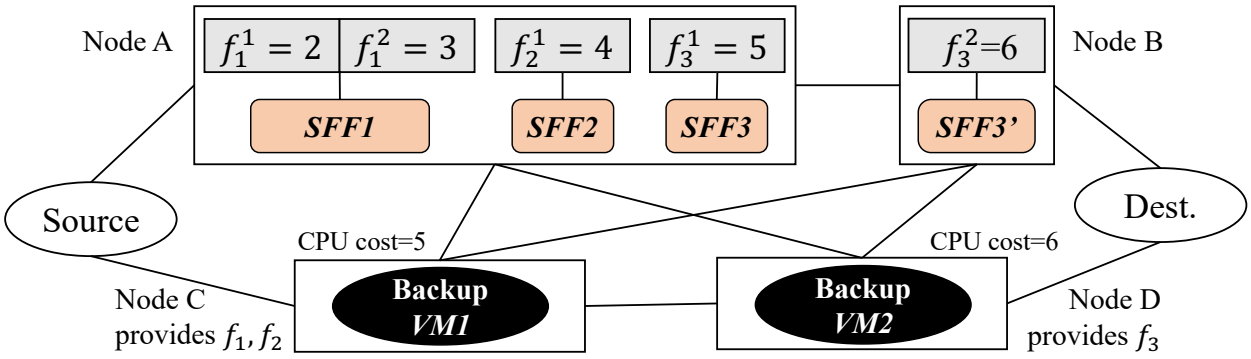


Figure 4.1: Service function forwarder (SFF) in NFV.

studies have focused on how to effectively handle the SFF failures. The traditional SF protection approaches protect either a single SF instance or multiple SF instances within a network by allocating enough backup SF instances. To install the backup SF instances, a variety of stochastic models [80, 81, 82] are applied depending on the available computing resources on the physical nodes and the sharing manner. However, the failure of an SFF can affect multiple SF instances of different SFCs connected to it simultaneously in an unexpected way [83].

We use Fig. 4.1 to illustrate the impact of an SFF failure, which includes a PN with four deployed SFFs. In this figure, a grey box denotes a running SF instance, a black ellipse denotes a backup virtual machine (VM), and an orange box denotes an SFF. Four running SF instances (i.e., $f_1^1, f_1^2, f_2^1, f_3^1$) are instantiated in Node A, whereas f_1^1, f_1^2 connects to $SFF1$, f_2^1 connects to $SFF2$, and f_3^1 connects to $SFF3$. In Node A, two instances f_1^1 and f_1^2 share $SFF1$ and require 2 and 3 CPUs[1], respectively. The instance f_2^1 demands 4 CPUs, while the instance f_3^1 demands 5 CPUs. Node B has one instance f_3^2 demands 6 CPUs and connects

[1]Without loss of generality, we use the number of CPUs representing the resource demands by SF instances.

to $SFF3'$. Node C has no running SF instances, but can host instances of f_1 and f_2. Any misconfiguration or security attack in $SFF1$ on Node A will cause simultaneous failures of f_1^1 and f_1^2 [72], which is called type-oriented SFF failures. In such a failure scenario, one would have to identify backup VMs running f_1 to simultaneously protect f_1^1 and f_1^2. As Node C provides f_1 and off-site protection, backup $VM1$ can protect f_1^1 and f_1^2 by initializing the corresponding instances with $2 + 3 = 5$ CPUs.

It is imperative that the SFFs (in conjunctions with the connected SF instances) are simultaneously protected. As a result, the backup resources on backup nodes are no longer able to be shared or assigned randomly between SF instances that are connected to the same SFF, which makes it infeasible to directly apply the traditional SF protection approaches (e.g., [80]). In this work, we investigate how to efficiently allocate backup resources for SF instances in a shared PN against the type-oriented SFF failure, which is called off-site type-oriented SFF failure protection (TSFP). To optimize TSFP, we present an efficient heuristic approach and derive its logarithm-approximation bound. Experiments show that our proposed algorithm outperforms approaches derived from the existing work.

The rest of this chapter is organized as follows. In Section 4.2, we formulate the TSFP problem with a mathematical model. Section 4.3 presents the type-oriented off-site shared protection optimization (TOPO) algorithm. In Section 4.4, we analyze and evaluate our proposed schemes. Finally, we conclude this chapter in Section 4.5.

4.2 Off-site Type-oriented SFF Failure Protection (TSFP)

4.2.1 Physical Network

Consider a physical network (PN) as an undirected graph denoted by $PN = (N, L)$, where N is the set of physical nodes, L is the set of physical links. Each physical node $n \in N$ represents a server or data center, which has the total computing capacity $\mathbb{C}_n$. For each physical link $l_{m,n} \in L$, where m and n (i.e., $\forall m, n \in N, m \neq n$) are its endpoints, it has a specific amount of latency $E_{l_{m,n}}$ and bandwidth $BW_{l_{m,n}}$. A physical path $P_{m,n}$ is composed of multiple physical links from Node m to n, which is the shortest path between m and n. We use $E_{P_{m,n}}$ and $BW_{P_{m,n}}$ to represent the latency and available bandwidth of $P_{m,n}$, respectively.

4.2.2 Problem Formulation

Given a PN with embedded SF instances, in which each physical node hosting limited types of SF connected to the SFFs, we investigate how to protect SF instances in a PN against type-oriented SFF failures with minimum amount of overall backup resources consumption. We name this problem as the off-site type-oriented SFF failure protection (TSFP). The off-site requirement enforces that the backup VMs must be in a physical node other than the node hosting the failed SFF, which is critical in handling failures due to natural disasters or deny-of-service (DOS) attacks.

To against the type-oriented SFF failure, backup nodes must be identified to protect all SF instances connected to the SFF. For example, in Fig. 4.1, the resource demands of f_1^a

and f_1^b connected to $SFF1$ have to simultaneous be protected. When there is just one type-oriented SFF failure, backup resources from the backup node can be shared among different SFFs. We use ξ_n^i to represent a set of SF instances connected to i^{th} SFF in Node n. In Fig.4.1, $\xi_A^1 = \{f_1^1, f_1^2\}$, $\xi_A^2 = \{f_2^1\}$, $\xi_A^3 = \{f_3^1\}$ and $\xi_B^3 = \{f_3^2\}$. Let $\mathbb{C}_{\xi_n^i}$ denote the resource demands of ξ_n^i, as shown in Eq. (4.1), where $\mathbb{C}_v$ is the resource demands of SF instance v. Table I lists the notations.

$$\mathbb{C}_{\xi_n^i} = \sum_{v \in \xi_n^i} \mathbb{C}_v \tag{4.1}$$

Eq. (4.2) shows the objective function of minimizing overall backup resource consumption in the PN.

$$\min \sum_{\forall m \in N} \mathbb{B}_m \tag{4.2}$$

Eq. (4.3) defines a binary variable $\chi_{m,n}^i, m \in N, n \in N, m \neq n, i \in \mathbb{F}_n$, where $\mathbb{F}_n$ is a set of SFFs in Node $n \in N$. $\chi_{m,n}^i$ equals 1 if Node m provides protection for ξ_n^i in Node n, and 0 otherwise.

$$\chi_{m,n}^i = \begin{cases} 1, & \text{if } m \text{ provides protection for } \xi_n^i \text{ in } n. \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

For the off-site backup requirements, Eq. (4.4) ensures that each SF instance in Node n can only be off-site backed up by another Node m (i.e., $m \neq n$). Eq. (4.5) restricts that each

Table 4.1: Notations.

Notation	Meaning
N	The set of physical nodes in the PN.
L	The set of physical links in the PN.
m, n	A physical node in N.
$l_{m,n}$	A physical link between m and n.
$E_{l_{m,n}}$	The latency on $l_{m,n}$.
$BW_{l_{m,n}}$	The bandwidth on $l_{m,n}$.
$P_{m,n}$	A physical path between m and n.
$E_{P_{m,n}}$	The latency on $P_{m,n}$.
$BW_{P_{m,n}}$	The bandwidth on $P_{m,n}$.
v	An SF instance running specific SF.
$\mathbb{F}_n$	A set of SFFs in Node n.
ξ_n^i	a set of SF instances connected to i^{th} SFF in Node n.
$\mathbb{C}_{\xi_n^i}$	The resource demands of ξ_n^i.
$\mathbb{C}_v$	The resource demands of SF instance v.
$\mathbb{C}_m$	The total computing capacity of Node m.
$\mathbb{B}_m$	The backup resource consumption of Node m
$\chi_{m,n}^i$	A binary variable to determine whether Node m provides protection for ξ_n^i in Node n.
$\gamma_{m,n}^i$	A binary variable to determine if $P_{m,n}$ is employed to backup traffic from ξ_n^i to m.
Θ	The latency threshold.
BW	The required bandwidth resources for backup.

physical node cannot protect the SF instances hosted by itself.

$$\sum_{m \in N, m \neq n} \chi_{m,n}^i = 1, \quad \forall i \in \mathbb{F}_n, \forall n \in N \tag{4.4}$$

$$\chi^i_{m,n} = 0, \quad \forall i \in \mathbb{F}_n, \forall m, n \in N, m = n \tag{4.5}$$

We use $\mathbb{B}_m$ to represent the overall backup resource consumption of Node m, which is calculated by Eq. (4.6).

$$\mathbb{B}_m = \max_{i \in \mathbb{F}_n, n \in N} \mathbb{C}_{\xi^i_m} * \chi^i_{m,n}, \quad \forall m \in N \tag{4.6}$$

Eq. (4.7) is the constraint of computing resources at a physical node.

$$\mathbb{B}_m \leq \mathbb{C}_m - \sum_{i \in \mathbb{F}_m} \mathbb{C}_{\xi^i_m}, \ \forall m \in N \tag{4.7}$$

Let $\gamma^i_{m,n}, m \in N, n \in N, m \neq n, i \in \mathbb{F}_n$ represent a binary variable; $\gamma^i_{m,n}$ equals 1 if $P_{m,n}$ is a backup path to transmit the backup traffic from ξ^i_n in Node n to Node m; 0 otherwise.

$$\gamma^i_{m,n} = \begin{cases} 1, & \text{if } P_{m,n} \text{ is employed to backup traffic} \\ & \text{from } \xi^i_n \text{ to } m. \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

The required bandwidth resources for backup is denoted by BW, Eq. (4.9) ensures that if $P_{m,n}$ is employed to backup traffic from ξ^i_n to Node m, then $P_{m,n}$ must have enough bandwidth. Let Θ denote the latency threshold, Eq. (4.10) ensures that Node m must be not far away from Node n for backup.

$$\sum_{m \in N} \sum_{n \in N} \gamma^i_{m,n} * BW \leq BW_{P_{m,n}}, \quad \forall i \in \mathbb{F}_n \tag{4.9}$$

$$E_{P_{m,n}} * \gamma_{m,n}^{i} \leq \Theta, \quad \forall i \in \mathbb{F}_n, \forall m, n \in N, m \neq n \tag{4.10}$$

Note that, when each physical node hosts limited SFs in an edge network, by relaxing the off-site backup constraints, the TSFP problem can be reduced to the minimum weighted set cover (MWSC) problem, which is NP-hard [84].

4.3 Type-oriented Off-site Shared Protection Optimization (TOPO)

In this section, we introduce the type-oriented gain (TG) to identify suitable backup candidates, and then propose a novel heuristic approach called type-oriented off-site shared protection optimization (TOPO).

According to Eqs. (4.4) and (4.5), to provide off-site protection, one can greedily pick a backup node by applying the traditional physical node or SF instance protection methods (e.g., largest degree first, or shortest path first). However, these methods may not optimize the backup resource consumption, as they cannot effectively take advantages of the sharing among different SFFs. From our observation, the number of backup nodes increases when the backup resource consumption increases for the type-oriented SFF failure. A backup node may use less backup resources when protecting as many SFFs as possible across different physical nodes by sharing the same amount of backup resources. Let $|\mathbb{S}_m|$ denote the cardinality of the SF instances from identical SFFs that are protected by backup node m, and $\mathbb{B}_m$ is its backup resource consumption. A type-oriented gain (TG) can be defined for each of the potential backup node to measure how backup resources can be effectively utilized by SF

instances from identical SFFs, as shown in Eq. (4.11). Here, the higher TG value indicates the more SF instances from the SFFs can be protected by using less backup resources due to the sharing among different SFFs.

$$TG_m = \max \frac{|\mathbb{S}_m|}{\mathbb{B}_m} \tag{4.11}$$

Following that, we propose the TOPO algorithm as shown in Alg. 5. To begin with, TOPO discards all physical links that violate the routing constraints associated with bandwidth and latency requirements in a PN based on Eqs. (4.8)-(4.10). Thereafter, TOPO creates three sets: the backup node set BN as empty, the backup candidate set BC as all physical nodes, and the unprotected SF instance set UF as all SF instances. TOPO repeats the following operations until UF becomes empty: (i) calculates and updates the TG value for each element in BC based on Eq. (4.11), (ii) sorts BC based on the TG value, and selects node $x \in BC$ which has the highest TG value as the backup node, (iii) identifies the protected SF instances set $\mathbb{S}_x$ for node x, (iv) updates the backup cost, and updates UF by removing $\mathbb{S}_x$, (v) moves backup node x from BC into BN. In the end, the TOPO algorithm returns the backup node set BN and the overall backup resource consumption *cost*.

As one can see that, Alg. 5 takes $\mathcal{O}(|N| + |L|)$ to discard the invalid physical links in the PN. The calculation of the TG value depends on the number of SF instances with identical SFFs that can be protected by the backup nodes and the required computing resources from these SF instances. Suppose there are at most $|V|$ SF instances on a physical node, and it takes at most $|N|$ times to traverse all backup candidate nodes, so the time complexity of calculating the TG value of each backup candidate node is $|NV|$. After

Algorithm 5 Type-oriented off-site shared protection optimization (TOPO) Algorithm

Input: G';
Output: BS, *cost*;
Discard physical links that violate Eqs. (4.9) and (4.10);
Initialize $BN = \emptyset$, $BC = N$, and UF as all SF instances;
while $UF \neq \emptyset$ **do**
    Calculate and update the TG value for each element in BC based on Eq. (4.11);
    Sort BC based on the TG value, and select node x that has the highest TG value as the backup node;
    Identify the protected SF instances set $\mathbb{S}_x$ for node x;
    Update the backup cost, and $UF = UF - \mathbb{S}_x$;
    Move backup node x from BC into BN;
end while
return BN, *cost*;

that, it takes at most $|N|$ iterations to find all backup nodes such that all SF instances in the PNs are protected. As a result, in the worst case, TOPO has the time complexity as $\mathcal{O}(|N| + |L| + |N^2V|) = \mathcal{O}(|N^2V|)$.

Table 4.2: Notations.

Notation	Meaning
$\lvert\mathbb{S}_{\delta_i}\rvert$	The number of SF instances protected by TOPO.
δ_i	The backup candidate node selected in i^{th} iteration.
$\lvert UF_i\rvert$	The unprotected SF instances in i^{th} iteration.
BN_i^{OPT}	The optimal backup node set in i^{th} iteration.
$\lvert\phi_i\rvert$	The number of backup nodes in BN_i^{OPT}.
$cost_{\text{TOPO}}$	The overall backup resource consumption of TOPO.

Theorem 4.3.1. *When each physical node hosts limited required SFs, the TOPO algorithm is logarithm-approximate.*

Proof. Let $|\mathbb{S}_{\delta_i}|$ represent the number of SF instances protected by TOPO, and δ_i represents

the backup candidate node selected in i^{th} iteration. To facilitate the presentation for the bound proof, the necessary notations are given in Table II. Eq. (4.12) shows the relationship between the number of protected SF instances and unprotected SF instances in adjacent iterations (i.e., iterations i and $i+1$).

$$|UF_i| = |\mathbb{S}_{\delta_i}| + |UF_{i+1}| \tag{4.12}$$

As the TOPO algorithm picks the backup candidate node in BC with the highest TG value as a backup node, TG_{δ_i} should be larger than the average TG value of the backup nodes from the optimal solution. Let BN_i^{OPT} represent the optimal backup node set in i^{th} iteration, and $|\phi_i|$ denote the number of backup nodes in BN_i^{OPT}, then Eq. (4.13) holds.

$$\frac{|\mathbb{S}_{\delta_i}|}{\mathbb{B}_{\delta_i}} \geq \frac{\sum_{m \in BN_i^{OPT}} \frac{|\mathbb{S}_m|}{\mathbb{B}_m}}{|\phi_i|} \tag{4.13}$$

To maximize the TG value of the backup node m in BN_i^{OPT}, its backup resources should not be greater than the backup resources of BN_i^{OPT} as shown in Eq. (4.14).

$$\mathbb{B}_m \leq \mathbb{B}_{BN_i^{OPT}} \tag{4.14}$$

Following that, identical SFFs can be shared among different physical nodes, Eq. (4.15) holds.

$$\sum_{m \in BN_i^{OPT}} |\mathbb{S}_m| \geq |UF_i| \tag{4.15}$$

Combining Eqs. (4.13)-(4.15), we have Eq. (4.16) as following.

$$\mathbb{B}_{\delta_i} \leq |\mathbb{S}_{\delta_i}| * \frac{\mathbb{B}_{BN_i^{OPT}} * |\phi_i|}{|UF_i| * |\phi_i|} \tag{4.16}$$

Next, we can derive Eq. (4.17) from Eq. (4.12).

$$\frac{|\mathbb{S}_{\delta_i}|}{|UF_i|} \leq \frac{1}{|UF_i|} + \frac{1}{|UF_i|-1} + \ldots + \frac{1}{|UF_{i+1}|} \leq \sum_{\omega=|UF_{i+1}|}^{|UF_i|} \frac{1}{\omega} \tag{4.17}$$

Assuming the proposed TOPO algorithm has a total of I iterations to select the proper backup nodes for the entire PN, we combine Eqs. (4.16) and (4.17) to Eq. (4.18) as follows.

$$\sum_{i=1}^{|I|} \mathbb{B}_{\delta_i} \leq \sum_{\omega=|UF_{i+1}|}^{|UF_i|} \frac{1}{\omega} * \mathbb{B}_{BN_i^{OPT}} \leq \sum_{\omega=|UF_1|}^{|UF_{|I|}|} \frac{1}{\omega} * \mathbb{B}_{BN^{OPT}} \tag{4.18}$$

Since Eq. (4.18) follows the properties of the harmonic series, it can be transformed into Eq. (4.19), where UF_0 represents the unprotected SF instances in the 0^{th} iteration and equals to the total number of SF instances in the PN.

$$\begin{aligned} \sum_{i=1}^{|I|} \frac{1}{\omega} &= \frac{1}{|UF_0|} + \frac{1}{|UF_0| - |\mathbb{S}_{\delta_1}|} + \ldots + \frac{1}{|\mathbb{S}_{\delta_{|I|}}|} \\ &\leq \frac{1}{|UF_0|} + \frac{1}{|UF_0| - k} + \ldots + 1 \leq \frac{ln(|UF_0|)}{k} + 1 \end{aligned} \tag{4.19}$$

Let $cost_{\text{TOPO}}$ denote the overall backup resource consumption of TOPO. Eq. (4.19) can be extended as the following.

$$cost_{\text{TOPO}} = \sum_{i=1}^{|I|} \mathbb{B}_{\delta_i} \leq (\frac{ln(|UF_0|)}{k} + 1) * \mathbb{B}_{BN^{OPT}} \tag{4.20}$$

Thus, the proposed TOPO algorithm achieves a logarithm-approximation performance. □

4.4 Experimental Results

4.4.1 Simulation Settings

To evaluate the performance of the proposed TOPO algorithm, we conduct our experiments in 24-nodes-43-links mesh networks. Each physical link has a bandwidth set in the range of $[5, 20]$. The latency of each physical link is set in the range of $[10, 30]$. The computing capacity of each physical node is set in the range of $[100, 400]$. Each physical node hosts four random SFs and equips with four SFFs. Note that every SF will initialize $[1, 4]$ SF instances in the PN. The resource demands of each SF instance are set in the rang of $[10, 40]$.

We implement two greedy algorithms called internal sharing gain only (ISGO) and external sharing gain only (ESGO) algorithms to evaluate the sharing efficiency of the proposed TOPO. In specific, ISGO only allows SFF sharing in the physical node, and no sharing is allowed among different physical nodes. Similarly, ESGO only allows SFF sharing in the same type of i^{th} SFF, and no sharing is allowed among different type of SFFs. Furthermore, to compare the performance of the proposed TOPO algorithm with the state-of-the-art techniques, we implement the minimum cost pairwise node assistance (NOTION) algorithm in [80]. We extend NOTION by 1) identifying a pair of physical nodes that have most common SFFs, 2) selecting this node pair to protect the identical SFFs on each other by only sharing backup resources to the SFFs inside of each other first, then protecting as many SFFs on the rest of physical nodes as possible, and 3) repeating the previous step until all SFFs are protected.

4.4.2 Performance Analysis

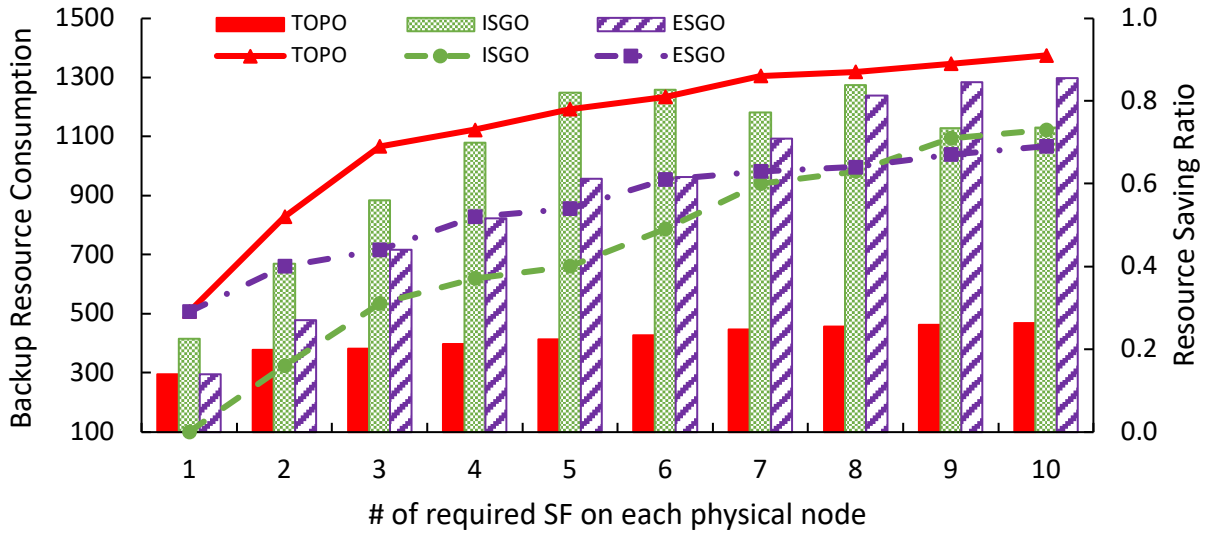


Figure 4.2: Performance comparison of TOPO, ISGO and ESGO.

Fig. 4.2 shows the backup resource consumption and the resource saving ratio of TOPO, ISGO, and ESGO when increasing the number of required SF on each physical node. Here, the resource saving ratio is the proportion of resources saved through sharing to the total amount of resources that must be backed up. In Fig. 4.2, the red solid bar (solid curve), green dotted bar (dashed curve), and purple stripes bar (dash-dotted curve) represent the performance of TOPO, ISGO, and ESGO, respectively. The bars are measured by the left y-axis, and the curves are measured by the right y-axis.

When increasing the number of required SF, more SF instances need to be backed up, resulting in all protection schemes consume more backup resources. Specifically, TOPO uses less backup resources than the minimum backup resources used by either ISGO or ESGO. This is because Eq. (4.11) ensures the proposed TOPO to minimize the needed backup resources by applying the SFF sharing. Fig. 4.3 illustrates an example of how the backup

resources are allocated by TOPO for the PN in Fig. 4.1. The overall backup resource consumption of TOPO is 11 CPUs. However, when applying ISGO in Fig. 4.1, Node D cannot share the backup resources among $SFF3$ in Node A and B, which has to provide 5+6=11 CPUs to backup them, thereby Node C and D cost 5+11=16 CPUs in Fig. 4.1. Similarly, when applying ESGO in Fig. 4.1, Node C cannot share any backup resources for $SFF1$ and $SFF2$ inside of Node A, which has to provide 5+4=9 CPUs to backup them, and the overall backup resource consumption is 9+6=15 CPUs in Fig. 4.1. As a result, TOPO significantly outperforms ISGO and ESGO.

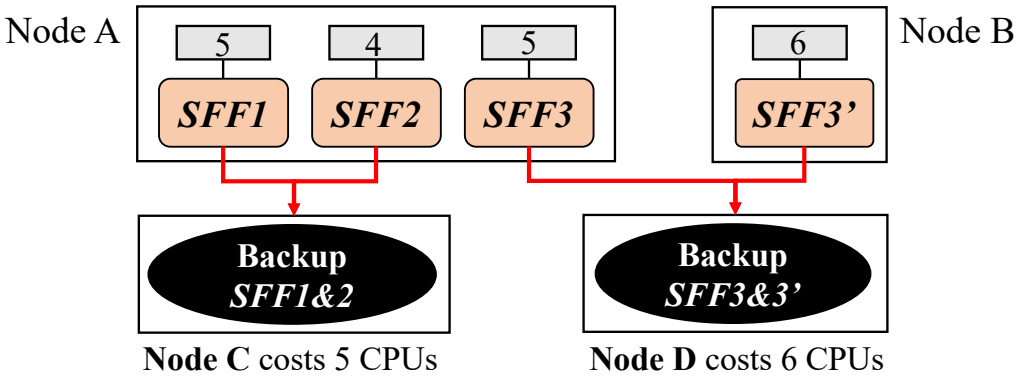


Figure 4.3: Backup resource allocation of TOPO.

In Fig. 4.2, when there is only one required SF on each physical node, the resource saving ratio of ISGO is 0. This is because when only one SFF inside a physical node capable of running SF instances, the backup node cannot provide any SFF sharing. For example, when applying ISGO in Fig. 4.3, Node D cannot provide any SFF sharing for $SFF3$ in Node A or B. Due to the same reason, we can see that ESGO and TOPO achieve the same performance when there is no SFF sharing for the case of only one required SF on each physical node.

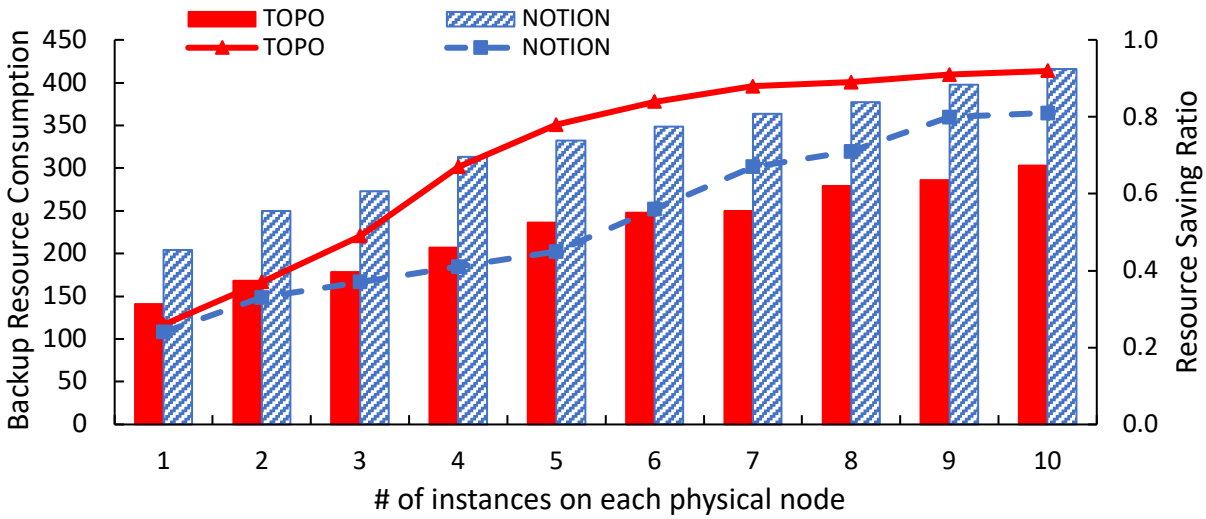


Figure 4.4: Impact of the number of SF instances on each physical node.

Furthermore, we observe that ESGO outperforms ISGO when the number of required SF on each physical node is less than 7. However, when further increasing the number of SFs, ISGO outperforms ESGO. This can be explained as follows. When the number of required SF on each physical node is small, there is not many SFFs per node, and the SFF sharing opportunity is small. In this case, the SFF sharing across different physical nodes will dominate. However, with more SFs hosted by each physical node, more SFFs will increase the chances for sharing inside of physical nodes.

Figs. 4.4-4.6 show the performance comparison among the proposed TOPO and NOTION, where the red solid bar (solid curve) and blue stripes bar (dashed curve) represent the performance of TOPO and NOTION, respectively. The bars are measured by the left y-axis, and the curves are measured by the right y-axis. In Fig. 4.4, the required backup resources increase with the number of SF instances as more running SF instances demand more resources. We can also see that TOPO outperforms NOTION by as much as 30%.

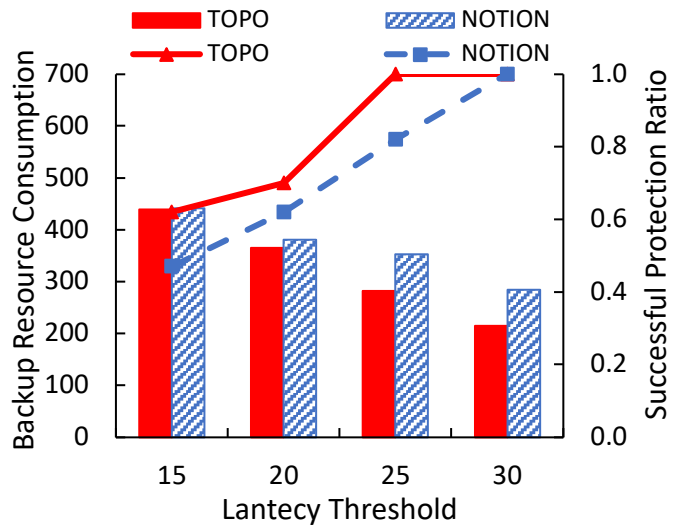


Figure 4.5: Impact of latency threshold.

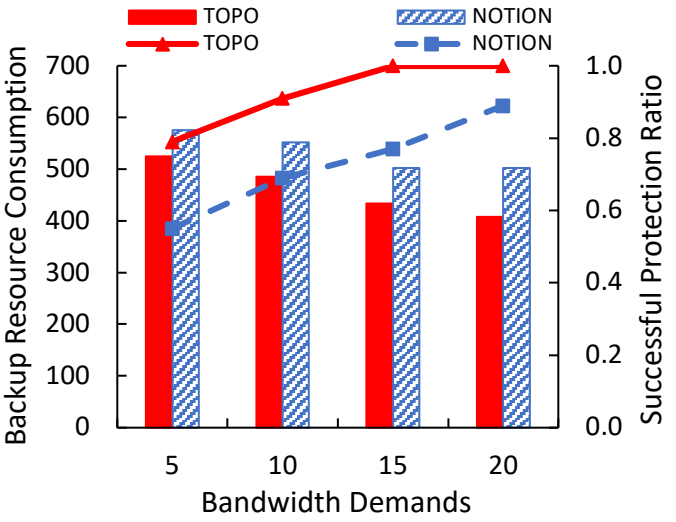


Figure 4.6: Impact of bandwidth demands.

This is because the proposed TOPO can take advantages of sharing the SFFs within physical nodes and cross different physical nodes concurrently in Eq. (4.11), leading to less backup resources protect more SF instances. However, NOTION exploits the SFF sharing inside of physical nodes, and cross different physical nodes in a sequential manner.

Fig. 4.5 shows the backup resource consumption and the successful protection ratio (i.e.,

the ratio between the number of protected SF instances and the number of all SF instances) when increasing the latency threshold. All approaches require less backup resources to backup all SF instances against type-oriented SFF failure. TOPO outperforms NOTION by using less backup resources, because with a larger given latency threshold, each physical node can protect more SF instances in other physical nodes. Fig. 4.6 shows the performance of all schemes when varying the available bandwidth. All schemes have higher successful protection ratios when increasing expanding the bandwidth in each physical link. When the available bandwidth is less, the number of candidate backup nodes is limited by the available bandwidth, leading to less chance of sharing backup resources. On the average, TOPO outperforms NOTION by as much as 20%.

4.5 Summary

In this chapter, we have studied a shared backup resource allocation problem to defend the type-oriented SFF failures. We have defined a novel problem called the off-site type-oriented SFF failure protection (TSFP) in NFV, and mathematically formulated this problem. We have proposed an efficient heuristic algorithm called type-oriented off-site shared protection optimization (TOPO) and proved its logarithm-approximation. Extensive simulation results have verified that the proposed TOPO outperforms the approaches directly extended from the existing work.

CHAPTER 5

SERVICE FUNCTION FORWARDER FAILURE AND PROTECTION IN EDGE NETWORKS

5.1 Motivation

With the widespread deployment of 5G infrastructure, many internet applications require ultra-reliable and low-latency communication (URLLC) services [85, 86, 87]. To deliver URLLC services, traditional middleboxes can be implemented to provide network functions, such as firewall, network address translation (NAT), and load balancers (LD), etc [68]. However, these proprietary middleboxes introduce high capital expenditures (CAPEX) and operational expenses (OPEX) [82]. To deliver the high-performance URLLC services with significant flexibility, scalability, and adaptability, network function virtualization (NFV) is introduced to replace the proprietary middleboxes with software-based modules, called service functions (SFs) [88]. In the NFV paradigm, SFs can be realized through virtual machines running on the commodity servers to process clients' traffic data [80]. A client's service request (SR) is generally comprised of multiple SFs and corresponding resource demands (e.g., CPU, bandwidth, virtual machines, etc.) [13]. To deploy an SR, service providers pack the required SFs into a service function chain (SFC) and embed it onto a shared physical network (PN) [21]. The process of accommodating an SR in a shared PN is referred to as service function chaining and embedding (SFCE) [43]. The SFCE process will yield a physical path for the SFC, which is called a service function path (SFP). For example, as shown in Fig. 5.1, there are two SFCs (i.e., $SFC1$ and $SFC2$) and the corresponding SFPs

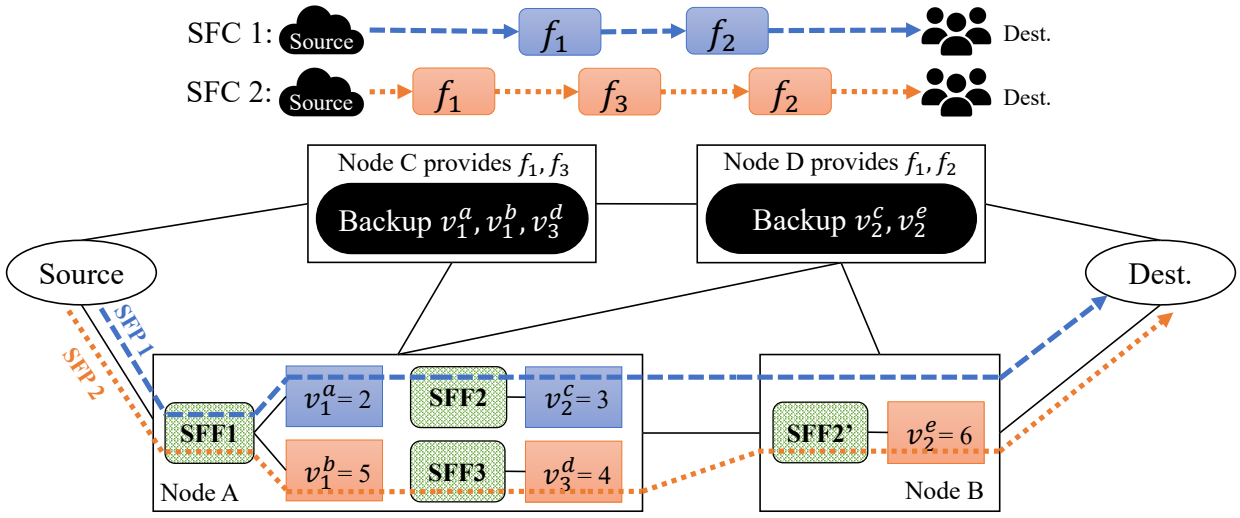


Figure 5.1: Example of a PN with four SFFs.

are $SFP1$ and $SFP2$. A service function forwarder (SFF) is in charge of forwarding traffic along the SFP [89]. There may be various running SF instances connect to the same SFF, and these SF instances may belong to separate SFPs, sharing the database or networking resources [72].

For URLLC applications such as the remote surgery and intelligent transportation systems, SFP protection strategies must be provided to recover from diverse network failures [90]. In the literature, many studies have focused on how to protect against the failures from physical nodes, links, and SF instances. For example, the work in [46] focused on protecting physical node failures. The work in [59] investigated how to recover from physical link failures and [78] explored an efficient backup approach to recover from SF instance failures.

In this work, we study a new type of failure: service function forwarder failure in NFV. Fig. 5.1 shows an example of a PN with four SFFs, where the four white boxes are physical

nodes (e.g., edge servers). In each physical node, the green box denotes an SFF and the black box denotes a backup SF instance. In Node A, SF instances v_1^a and v_1^b are running the same SF (i.e., f_1) via the same SFF (i.e., $SFF1$). The SF instance v_1^a and v_1^b demand 2 and 5 CPUs[1] to run SF f_1, respectively. The v_2^c requires 3 CPUs to run f_2 through the connection of $SFF2$, and v_3^d requires 4 CPUs to run f_3 through $SFF3$. Similarly, in Node B, v_2^e requires 6 CPUs to run f_2 through $SFF2'$. Node C has no actively running SF instances, but can support f_1 and f_3. That is to say, one can install backup SF instances running f_1 or f_3 when SF instance failures (e.g., v_1^a or v_3^d) occur. Similarly, Node D can run backup SF instances for f_1 and f_2. In Fig. 5.1, $SFC1$ ($Source \rightarrow f_1 \rightarrow f_2 \rightarrow Dest.$) is delivered through the blue-dashed SFP ($Source \rightarrow v_1^a \rightarrow v_2^c \rightarrow Dest.$), while $SFC2$ ($Source \rightarrow f_1 \rightarrow f_3 \rightarrow f_2 \rightarrow Dest.$) is delivered through the orange-dotted SFP ($Source \rightarrow v_1^b \rightarrow v_3^d \rightarrow v_2^e \rightarrow Dest.$). When $SFF1$ in Node A fails, all connecting SF instances (i.e., v_1^a, v_1^b) will not work and the corresponding SFCs (i.e., $SFC1, SFC2$) will break down.

Defending against the SFF failure is challenging as multiple SF instances may be brought down due to one SFF failure. In Fig. 5.1, once $SFF1$ in Node A fails, one would have to identify backup SF instance running f_1 to protect v_1^a, v_1^b simultaneously. Furthermore, off-site protection for such failures is important for many clients when facing the failures from natural disasters, attacks, or security breaches [74]. In this case, Node C can install backup SF instances of f_1 to recover the workload of v_1^a, v_1^b by using $2 + 5 = 7$ CPUs. In fact, if $SFF1$ and $SFF3$ in Node A do not fail together, Node C can provide shared protection for

[1]Without loss of generality, we use the number of CPUs representing the resource demands by SF instances.

v_1^a, v_1^b and v_3^d by using $\max\{2+5, 4\} = 7$ CPUs. This sharing of several SF instances within a physical node is referred to as *SFF internal sharing*. Similarly, when $SFF2$ in Node A or $SFF2'$ in B is down, Node D can backup v_2^c or v_2^e by using $\max\{3, 6\} = 6$ CPUs. This sharing of several instances with the same type of SF among different physical nodes is called *SFF external sharing*. Without considering the aforementioned *SFF internal and external sharing*, existing approaches on physical node protection or SF instances protection cannot be applied efficiently to resolve such SFF failures and related protection issues. For example, neither Node C nor D can replace Node A directly by applying the physical node protection approaches [46]. Similarly, the single SF instance approaches [78] cannot be applied to protect v_1^a and v_1^b concurrently.

This study analyzes how to efficiently allocate off-site backup resources for SF instances against the SFF failure with various service level agreements (SLAs) or quality of service (QoS) needs. To reduce the overall backup resource consumption, we mathematically formulate a new problem called SFF-driven multi-instances failures and protection (SMFP) and prove its complexity. To optimize SMFP, we propose the SFF-centralized resource optimization (FCRO) algorithm, an efficient heuristic approach that builds on the proposed backup auxiliary transferring, backup cost-effectiveness selection, and adaptive fit backup techniques. Our analysis and simulation results show that the FCRO is effective and outperforms the schemes that are directly extended from the existing work.

The rest of this chapter is organized as follows. Section 5.2 summarizes the related work. Section 5.3 formulates the SMFP and analyzes its complexity. Section 5.4 presents the FCRO

algorithm. Section 5.5 analyzes the experimental results. Section 5.6 concludes this work.

5.2 Related Work

In this section, we review the related work on service function chaining and embedding (SFCE) and network failures in NFV.

5.2.1 Service Function Chaining and Embedding

The existing work on the SFCE problem can be broadly classified into two categories, depending on whether the clients specify the SFC or not [17]. When the complete order of the SFs required in an SFC is given, the SFCE problem can be seen as a special case of the virtual network embedding problem [53], or the virtual network placement problem [55]. When the order of SFs required in an SFC is not fully given, the service provider needs to compose an SFC and embed it onto the PN.

5.2.2 Network Failures in NFV

In the literature, significant efforts have been put into investigating the network protection. In Fig. 5.2, we broadly classify the existing work on network failures in NFV as (i) physical component failures, or (ii) virtual component (service function) failures.

Physical component failures: Various studies have been devoted to protect the the network from the physical component failures, such as link or node failures [91, 92, 93]. In [91], the authors proposed a scheme to place the primary and backup sensors and fiber optic sensing routes to protect against the physical link failures. In [92], the authors exploited

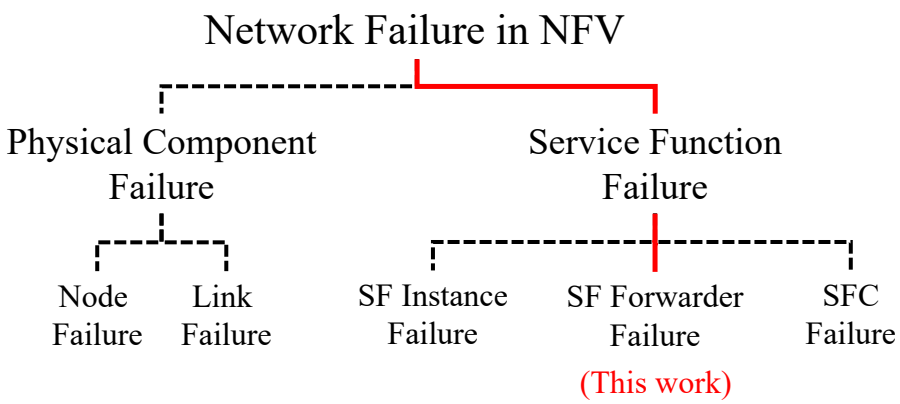


Figure 5.2: Classification of network failures in NFV.

a new virtual line-based multipath routing protocol to combat path failures. The authors in [93] studied a backup computing and transmission resource allocation model for virtual networks to handle multiple facility node failures.

Service function failures: Distinct from physical component failures that alter the topology and connectivity of a PN, the failures from the SF instance, SF forwarder and SFC can also result in severely service interruptions [72]. SF failures may come from the SF license expiration, misconfiguration, and software update errors and so on [94]. Considering the vulnerability of the SFC, the authors in [95] proposed an algorithm to efficiently place SFCs to enhance their availability. In [96], the authors explored the latency-aware and reliable SFC placement to improve the SFC reliability. Once an SF instance failure is detected, a pre-planned backup SF instance takes over and the traffic is redirected accordingly. In [78], the authors proposed an approximate approach to backup multiple SF instances in a latency-constrained system. The work in [62] considered a resource-aware backup allocation model to protect multiple SF instance failures by using the minimal backup resources.

5.2.3 Off-site Protection

When the primary SF instances fail, one can activate the pre-planned on-site backup SF instances to provide on-site protection locally [97]. However, these on-site backup SF instances remain in the same physical location as the primary SF instances, and they may connect to the same failed SFF. As a result, the backup SF instances may be rendered as unavailable in the event of SFF failures. In addition, when facing the failures from the natural disasters, attacks, or security breaches [74], the backup SF instances are better placed in a different (off-site) physical node to provide the protection [78, 80, 82]. Hence, in this chapter, we assume that the backup SF instances have to come from off-site physical nodes when SFF failures occur.

All connecting SF instances of an SFF have two sharing options: 1) *SFF internal sharing*, and 2) *SFF external sharing*. According to the example above, the SFF internal sharing allows an SFF to share backup resources with other SFFs of different SF types within a physical node, as long as they are all protected by the same backup node. Maximizing the SFF internal sharing efficiency of a backup node is equivalent to reducing the backup resource consumption. Intuitively, the more SFFs in a physical node protected by a backup node, the more backup resources can be saved through the SFF internal sharing. The SFF external sharing, on the other hand, allows an SFF to share backup resources with other SFFs of the same SF types on separate physical nodes.

Unlike a protection approach for the SF instance failure, the protection approach for the SFF failure must take all connecting SF instances through the same SFF into account

concurrently. Furthermore, how to effectively take advantages of the unique opportunities of SFF internal sharing and external sharing is still open and challenging. In the following sections, we present the SFF-driven multi-instance failure and protection (SMFP) problem and propose effective heuristics for the SMFP. In this chapter, we focus on utilizing both SFF internal and external sharing to improve SF instance reliability against the SFF failure. In the following section, we propose and study a new problem called SFF-driven multi-instances failures and protection (SMFP).

5.3 SFF-driven Multi-instances Failures and Protection (SMFP)

In this section, we present the mathematical formulation of the SFF-driven multi-instances failures and protection (SMFP) problem. Table 5.1 lists the notations.

5.3.1 Physical Network

Consider a physical network (PN) is an undirected graph denoted by $PN = (N, L)$, where N is the set of physical nodes, L is the set of physical links. Each physical node $n \in N$ can represent an edge server, it has some available computing resources denoted by c_n^R. Each physical node has a set of running SF instances denoted by E_n. Let $\delta_{i,n} \subseteq E_n$ represent the set of existing SF instances that are connected to the i-th SFF. For example, in Fig. 5.1, $\delta_{1A} = \{v_1^a, v_1^b\}$, $\delta_{2A} = \{v_2^c\}$, $\delta_{3A} = \{v_3^d\}$ and $\delta_{2B} = \{v_2^e\}$. Let $r_{i,n}$ represent the required computing resources of $\delta_{i,n}$, which is calculated by Eq. (5.1), where r_v is the

required computing resources of SF instance $v \in \delta_{i,n}$.

$$r_{i,n} = \sum_{v \in \delta_{i,n}} r_v \tag{5.1}$$

For each physical link $l_{m,n} \in L$, m and n (i.e., $\forall m, n \in N, m \neq n$) are its endpoints. A physical path $p_{m,n}$ is composed of a series of physical links from m to n, which has a specific amount of latency $la_{m,n}$ and available bandwidth $bw_{m,n}$.

5.3.2 Problem Formulation

Given a shared PN, each physical node has a number of SFFs connecting with a set of running SF instances. We investigate how to provide the SFF internal and external sharing protection for all existing SF instances, such that the total backup resource consumption is minimized while satisfying (i) SFF constraints, (ii) off-site protection constraints, (iii) SF capacity constraints, (iv) routing constraints, and (v) QoS latency constraints. We name this problem SFF-driven multi-instances failures and protection (SMFP) in NFV.

We formulate the SMFP problem as the following optimization problem to minimize the overall backup resource consumption. The objective function of the proposed SMFP problem is shown as Eq. (5.2), where c_m^B represents the allocated backup resources at Node m.

$$\min \sum_{\forall m \in N} c_m^B \tag{5.2}$$

SFF constraints: A backup node m can provide protection for the existing SF instances in $\delta_{i,n}$ only if it has the identical i-th SFFs as Node n. Let $x_m^{i,n}$ denote whether SF instances in $\delta_{i,n}$ on n is protected by m. In Eq. (5.3), $x_m^{i,n} = 1$ represents that the primary SF instances

Table 5.1: Notations.

Notation	Meaning
N	A set of physical nodes.
L	A set of physical links.
m, n	A physical node in N.
$\delta_{i,n}$	A set of SF instances connected to i-th SFF on n.
$r_{i,n}$	The required computing resources of $\delta_{i,n}$.
v	An SF instance in $\delta_{i,n}$.
r_v	The required computing resources of v.
E_n	The set of SF instances in n.
c_n^R	The available computing resources on n.
c_m^B	The allocated backup resources at m.
$l_{m,n}$	A physical link between m and n.
$p_{m,n}$	A physical path between m and n.
$la_{m,n}$	The latency on $p_{m,n}$.
$bw_{m,n}$	The bandwidth.
θ	The latency threshold on $p_{m,n}$
$\omega_{m,n}^v$	A binary variable to determine whether the backup path from m to n is employed to transmit traffic from the backup SF instance v' to the primary SF instance v.
$x_m^{i,n}$	A binary variable to determine whether SF instances in $\delta_{i,n}$ on n is protected by m.

in $\delta_{i,n}$ are protected by the backup node m; otherwise $x_m^{i,n} = 0$.

$$x_m^{i,n} = \begin{cases} 1, & \delta_{i,n} \text{ is protected by backup node } m. \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

Taking advantages of both the SFF internal and external sharing, a backup node m should allocate the largest required computing resources of $\delta_{i,n}$, and the backup resource consumption

of Node m can be expressed as Eq. (5.4).

$$c_m^B = \max_{n \in N, \delta_{i,n} \subseteq E_n} r_{i,n} * x_m^{i,n} \tag{5.4}$$

Off-site protection constraints: When SFF failures occur, the SF instances in $\delta_{i,n}$ have to be protected by another node m (i.e., $m \neq n$). Eq. (5.5) ensures that each SF instance in $\delta_{i,n}$ is protected by only one off-site backup node.

$$\sum_{\forall m \in N} x_m^{i,n} = 1, \quad \forall n \in N, m \neq n, \delta_{i,n} \subseteq E_n \tag{5.5}$$

SF capacity constraint: Since a set of primary SF instances running on each physical node, the allocated backup resources cannot exceed the available computing resources when it becomes a backup node offering the off-site protection.

$$c_m^B \leq c_m^R, \quad \forall m \in N \tag{5.6}$$

Routing constraints: When some SFFs fail, the connected SF instances have to be re-located to the corresponding backup nodes. As a result, the virtual links connected to the backup SF instances need to be remapped. Let $\omega_{m,n}^v$ indicate whether the backup path from the backup SF instance v' to the primary SF instance v is routed through $l_{m,n}$ as Eq. (5.7).

$$\omega_{m,n}^v = \begin{cases} 1, & \text{backup path from } m \text{ to } n \text{ employed to} \\ & \text{transmit traffic from } v' \text{ to } v. \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

QoS latency constraint: Eq. (5.8) restricts the backup node m cannot be far from Node n. The latency on the shortest distance between these two nodes cannot be bigger than the

latency threshold θ.

$$la_{m,n} * \omega^{v}_{m,n} \leq \theta, \quad \forall m, n \in N, m \neq n, \forall v \in E_n \tag{5.8}$$

5.3.3 Complexity Analysis

When each physical node can host all SFs from the clients' requests, any physical node m can be designated as the backup node, providing backup for the required SFs. By relaxing the stringent latency and bandwidth requirements, each physical node can provide SFF-based off-site shared protection for the other nodes when the topology of the PN is a complete graph. To satisfy the off-site protection constraints, Node m cannot protect existing SF instances on its own. The SF instances on Node m will be protected by any other Node n. As a result, the optimal number of backup nodes for the SMFP problem is two (i.e., Nodes m and n). Since a complete PN can have at most $|N^2|$ pairs of backup nodes, Theorem 5.3.1 holds.

Theorem 5.3.1. *When each physical node can host all SFs in a complete PN, the SMFP problem can be optimized within a polynomial time.*

Lemma 5.3.2. *If a given PN is a trivial graph such as circle, wheel, or star topology, the SMFP problem can also be solved within a polynomial time.*

Theorem 5.3.3. *When each physical node can host all SFs in a non-complete PN, the SMFP problem is NP-hard.*

Proof. We prove the NP-hardness of the SMFP problem in this scenario by showing a reduction from the minimum weighted set cover (MWSC) problem [84]. Given a finite universe

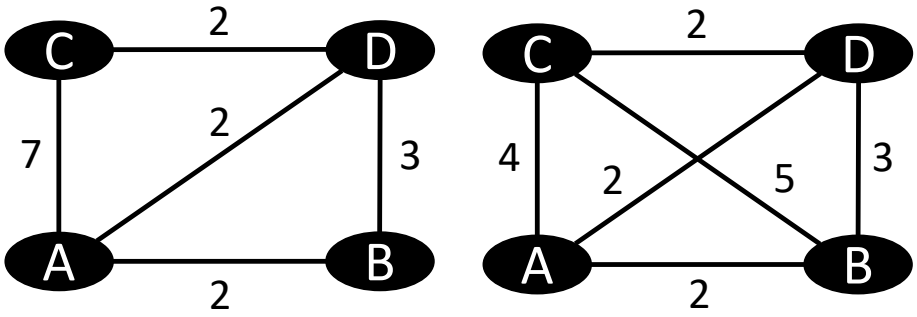
$U = \{u_1, u_2, ..., u_d\}$ of d elements, $S = \{s_1, s_2, ..., s_t\}$ is a collection of t subsets of U, and $W = \{w_1, ..., w_t\}$ assigns a positive real weight to each subset of U. The MWSC problem is a well-known NP-hard problem, which aims to find a set $I \subseteq S$, such that I covers all elements of U, and the sum of the weights of the subsets in I is minimized. Given an instance of the MWSC problem, we can construct an instance of the SMFP problem in a non-complete PN. First, we construct a bipartite graph based on the set cover instance: for each element $u_i \in U$, there is a node u_i in the left side of the bipartite graph, and for each subset $s_j \in S$, there is a node s_j in the right side of the bipartite graph. There is an edge $u_i s_j$ if and only if u_i belongs to subset s_j. By omitting the off-site protection constraint, the special case of the SMFP problem can be described as: given a set of unprotected SFFs that is $U = \{u_1, ..., u_d\}$, a set of potential backup nodes is $S = \{s_1, ..., s_t\}$, and a set of available positive real computing resources $W = \{w_1, ..., w_t\}$, how to find a set of backup nodes $I \subseteq S$ with the minimum backup resource consumption, such that each SFF $u_i \in U$ can be protected. It can be seen that a solution of the instance of the SMFP problem is optimal if and only if the corresponding solution of the MWSC instance is optimal. Therefore, in this scenario, the SMFP problem is NP-hard. □

5.4 SFF-centralized Resource Optimization (FCRO)

In this section, we first propose the novel backup auxiliary transferring (BAT), backup cost-effectiveness selection (BCS), and adaptive fit backup (AFB) techniques to facilitate the SMFP optimization. Then, we combine these techniques to propose our SFF-centralized resource optimization (FCRO) algorithm.

5.4.1 Backup Auxiliary Transferring (BAT)

As shown in Eq. (5.8), the URLLC services require strict QoS and SLA requirements, and not all physical links are involved in the backup process.



(a) A PN with latency. (b) A latency transferred PN'.

Figure 5.3: Example of BAT.

Fig. 5.3a shows an example of a PN with latency based on Fig. 5.1, where the number near each physical link represents its latency. Given a latency threshold $\theta = 5$, we identify that $l_{A,C}$ violates Eq. (5.8), and remove $l_{A,C}$ from PN. Then, we generate a new auxiliary graph PN' by connecting all nodes via the shortest path while satisfying the QoS latency constraint as shown in Fig. 5.3b. We call this process of generating a new latency transferred PN' as backup auxiliary transferring (BAT), and each auxiliary link in PN' is the shortest path between each pair of physical nodes.

5.4.2 Backup Cost-effectiveness Selection (BCS)

With the constraints of Eqs. (5.5) in Section IV, the SF instances in $\delta_{i,n}$ must be protected by another physical node. However, not all physical nodes can support the required SFs in

the PN as some of them may only host a limited set of SFs. A set of backup nodes with identical SFFs must be identified to protect the workload of $\delta_{i,n}$ in the PN. To minimize the overall backup resource consumption, it is critical to maximize the SFF internal and external sharing by identifying the appropriate backup nodes to maximize the sharing.

Intuitively, the SFF internal sharing efficiency is closely related to the number of protected SFFs within a physical node. The more SFFs in one physical node protected by a backup node, the more backup resources may be saved. Likewise, the SFF external sharing efficiency is closely related to the number of SFFs runs the same SF across different physical nodes. The more SFFs that runs the same SF, the more backup resources can be saved externally. Let $\mathbb{B}_m^{\text{internal}}$ represent a set of SFFs that can be protected by a potential backup node m through the SFF internal sharing, and $\mathbb{B}_m^{\text{external}}$ denotes a set of SFFs that can be protected by Node m through the SFF external sharing. The set of SFFs that can be protected by Node m through the SFF internal and external sharing is expressed by Eq. (5.9).

$$\mathbb{S}_m = \mathbb{B}_m^{\text{internal}} \cup \mathbb{B}_m^{\text{external}} \tag{5.9}$$

We present a backup cost-effectiveness selection (BCS) technique to ensure that the selected backup node can maximize the SFF internal and external sharing efficiency, as shown in Eq. (5.10).

$$\Gamma_m = \min \frac{c_m^B}{|\mathbb{S}_m \cap \mathbb{U}|} \tag{5.10}$$

In Eq. (5.10), we use $\mathbb{U}$ to represent the unprotected SFFs in the PN. The denominator of the fraction contains the actual number of SFFs that have been protected by node m in the

current iteration, whereas $\mathbb{S}_m$ includes SFFs that have been protected by previously selected backup nodes. In the numerator, c_m^B is the required backup resources to protect these SFFs. The smaller the value of Γ_m is, the more effective the potential backup node m will be in protecting the PN.

5.4.3 Adaptive Fit Backup (AFB)

Ideally, by employing the BCS technique, each selected backup node can maximize the benefits of the SFF internal and external sharing opportunities in each iteration. However, due to a lack of available computing resources to cover all connecting SF instances in $\delta_{i,n}$ (e.g., $r_{i,n} \geq c_m^R$), it is possible that no backup node can be identified for $\delta_{i,n}$. In this case, SF instances in $\delta_{i,n}$ have to be jointly protected by several backup nodes. We propose the adaptive fit backup (AFB) technique in Alg. 6 to identify the SF instances in $\delta_{i,n}$ that cannot be protected by a backup node as a whole. To begin with, for each physical node $m \in N$, AFB identifies all SFFs that satisfy the latency threshold θ and run the same SFs as the ones on Node m (except the SFFs on its own), and save them as $\mathbb{F}_m$. Then AFB identifies $\mathbb{S}_m$ according to Eq. (5.9). If $\mathbb{F}_m \neq \mathbb{S}_m$, it indicates that there must exist $\delta_{i,n}$ that cannot be protected by Node m as a whole. In this case, AFB identifies the potential backup nodes for each of $\delta_{i,n}$ in $\mathbb{F}_m \cap \mathbb{S}_m$, and save them as $\mathbb{P}(\delta_{i,n})$. Next, AFB assigns SF instances in $\delta_{i,n}$ separately to each $y \in \mathbb{P}(\delta_{i,n})$ until $\mathbb{F}_m \cap \mathbb{S}_m$ are protected. Lastly, AFB updates the backup resource consumption c_y^B of each Node $y \in \mathbb{P}(\delta_{i,n})$, and removes $\mathbb{F}_m \cap \mathbb{S}_m$ from $\mathbb{U}$.

For example, Fig. 5.4 shows the latency transferred graph PN' from Fig. 5.1. In Fig. 5.4, Nodes C and D require 4 CPUs to protect f_1, f_3, and 6 CPUs to protect f_1, f_2, respectively.

Algorithm 6 Adaptive fit backup (AFB)

Input: PN', and the resource allocation;
Output: c_y^B;
for each physical node $m \in N$ **do**
    Identify all off-site SFFs running the same SFs as the ones on m within θ, and save them as $\mathbb{F}_m$;
    Identify $\mathbb{S}_m$ according to Eq. (5.9)
    if $\mathbb{F}_m \neq \mathbb{S}_m$ **then**
        Identify all potential backup nodes for every $\delta_{i,n}$ in $\mathbb{F}_m \cap \mathbb{S}_m$, and save them as $\mathbb{P}(\delta_{i,n})$;
        Assign SF instances in $\delta_{i,n}$ separately to each $y \in \mathbb{P}(\delta_{i,n})$ until $\mathbb{F}_m \cap \mathbb{S}_m$ are protected;
        Update c_y^B and remove $\mathbb{F}_m \cap \mathbb{S}_m$ from $\mathbb{U}$;
    end if
end for
return c_y^B;

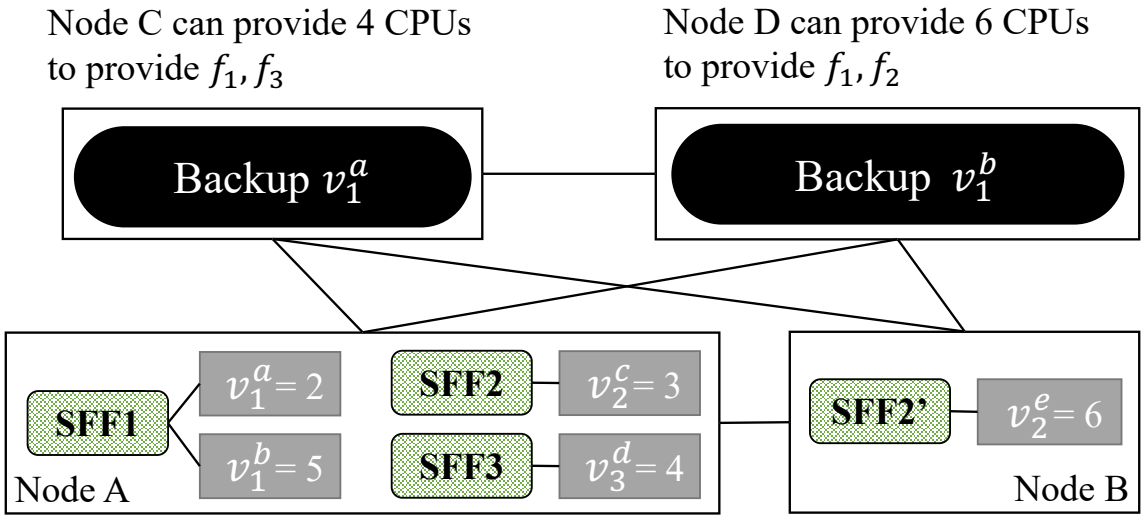


Figure 5.4: Example of AFB.

For each physical node in PN', we start with Node C to employ the AFB technique. To begin with, AFB identifies $\mathbb{F}_C = \{SFF3\}$, and $\mathbb{S}_C = \{SFF1, SFF3\}$. Since $\mathbb{F}_C \neq \mathbb{S}_C$, AFB identifies $\mathbb{F}_C \cap \mathbb{S}_C = \{SFF1\}$. Next, AFB identifies $\mathbb{P}(SFF1) = \{C, D\}$, and assigns the SF instances in $SFF1$ (i.e., v_1^a and v_1^b) separately to Nodes C and D. At last, AFB updates

Node C takes 4 CPUs and Node D takes 6 CPUs for backup, and removes $SFF1$ from $\mathbb{U}$.

5.4.4 SFF-centralized Resource Optimization (FCRO)

With the proposed BAT, BCS and AFB techniques, we introduce an efficient heuristic algorithm called SFF-centralized resource optimization (FCRO), as shown in Alg. 7. To begin with, FCRO initializes the backup resource consumption $cost = 0$, and creates a backup node set $\mathbb{B} = \emptyset$. Then, FCRO repeats the following steps until $\mathbb{U}$ is fully protected by $\mathbb{B}$ (Line 4-13): (i) calculates and updates Γ_m for each physical node $m \in N$ according to Eq. (5.10) (Line 5); (ii) picks the physical node x which has the minimum value of Γ_x as the backup node (Line 6); (iii) adds Node x into $\mathbb{B}$ and removes the protected SFFs from $\mathbb{U}$ (Line 7); (iv) updates the backup resource consumption $c_x^B = \max\{c_x^B, c_y^B\}$ if it is also the backup node selected in Alg. 6 (Line 8-11); and (v) updates the overall backup resource consumption of $\mathbb{B}$ (Line 12). At last, FCRO returns $\mathbb{B}$ and the corresponding backup resource consumption *cost* (Line 14).

Fig. 5.5 shows the flowchart of the proposed FCRO algorithm. In each iteration, FCRO selects the physical node with the smallest value of Γ_x indicating that the selected physical node achieved the smallest average backup resource consumption for each protected SFF. In addition, when the SF instances from $\delta_{i,n}$ are protected by the AFB technique separately, as long as the selected physical node in FCRO (i.e., Node x) is the one that selected in AFB (i.e., Node y), FCRO will remove the i^{th} SFF from $\mathbb{U}$, and update the backup resource consumption accordingly.

As one can see that, the calculation of Γ_m depends on the number of SFFs that can be

Algorithm 7 SFF-centralized Resource Optimization (FCRO)

Input: PN';
Output: $\mathbb{B}$, *cost*;
Initialize the backup resource consumption $cost = 0$, and create a backup node set $\mathbb{B} = \emptyset$;
while $\mathbb{U} \neq \emptyset$ **do**
    Calculate and update Γ_m for each physical node $m \in N$ according to Eq. (5.10);
    Pick Node x with the minimum value of Γ_x as the backup node;
    Add Node x into $\mathbb{B}$, remove protected SFFs from $\mathbb{U}$;
    if $x = y$ **then**
        Apply the AFB technique via Alg. 6 to get c_y^B;
        Update $c_x^B = \max\{c_x^B, c_y^B\}$;
    end if
    Update $cost = cost + c_x^B$;
end while
return $\mathbb{B}$, *cost*;

protected by each of physical node $m \in N$, and the required backup resource consumption of these SFFs. Suppose that there are at most $\mathbb{U}$ SFFs in the PN, and it takes at most $|N|$ times to traverse all physical nodes, the complexity of calculating Γ_m for each physical node is $|N * \mathbb{U}|$. Following that, it takes at most $|N|$ iterations to find all backup nodes such that all SFFs in the PN are protected. As a result, in the worse case, FCRO has the time complexity as $|N^2 * \mathbb{U}|$.

5.4.5 Boundary Analysis

In this section, we prove that the proposed FCRO algorithm achieve an H_d factor approximation. To facilitate the presentation for the bound proof, the necessary notations are given in Table 5.2.

Lemma 5.4.1. *Let $u_1, u_2, ...u_d$ denote the SFFs in $\mathbb{U}$, in the order in which they were protected. For each $k \in \{1, 2, ..., d\}$, $cost(u_k) \leq \frac{OPT}{d-k+1}$.*

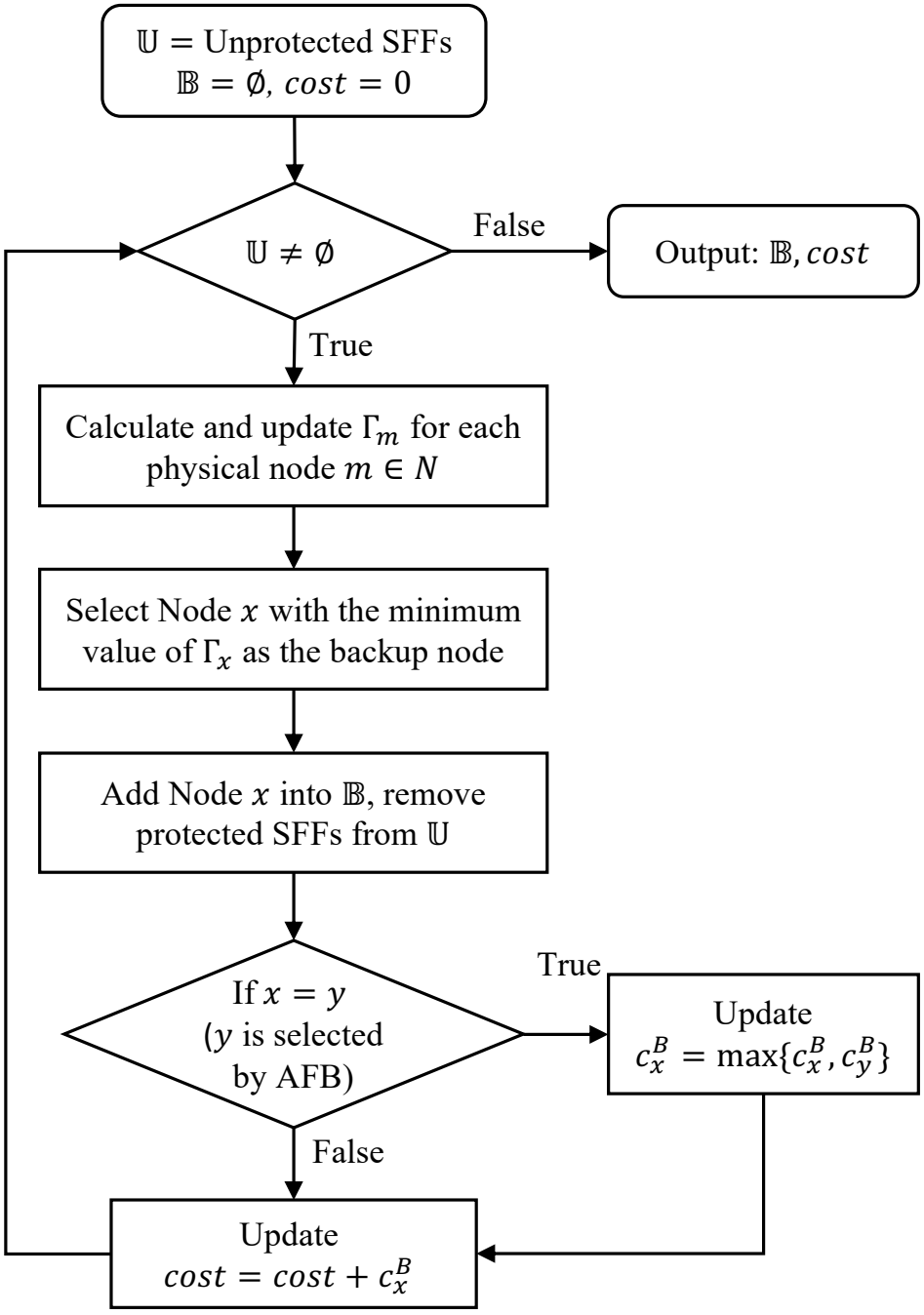


Figure 5.5: Flowchart of FCRO.

Table 5.2: Notations.

Notation	Meaning
$\mathbb{U}$	The set of unprotected SFFs in the PN.
d	The number of SFFs in $\mathbb{U}$.
u_k	The k-th SFF in $\mathbb{U}$ that is protected.
$cost(u_k)$	The average backup resource usage to protect u_k.
OPT	The optimal backup resource consumption by the optimal solution in any iteration.
α	The most cost-effectiveness backup node that is chosen in any iteration.
$\|\mathbb{S}^*\|$	The optimal number of SFFs protected by Node α through the SFF internal and external sharing in any iteration.

Proof. Let OPT denote the optimal backup resource consumption of protecting the remaining SFFs by an optimal solution in any iteration. It follows that there must exist at least one backup node α with $\Gamma_\alpha = \frac{OPT}{|\mathbb{S}^*|}$, where $|\mathbb{S}^*|$ is the optimal number of SFFs protected by Node α through SFF internal and external sharing. When u_k is protected, the size of $\mathbb{S}^*$ can be expressed as $d - k + 1 \leq |\mathbb{S}^*|$, where $1 \leq k \leq n$. Since u_k is protected by the most cost-effectiveness backup node α in this iteration, hence, we have Eq. (5.11) as follows.

$$cost(u_k) \leq \frac{OPT}{|\mathbb{S}^*|} \leq \frac{OPT}{d - k + 1} \tag{5.11}$$

□

Theorem 5.4.2. *The proposed FCRO algorithm can achieve an H_d factor approximation, where $H_d = 1 + \frac{1}{2} + ... + \frac{1}{d}$.*

Proof. According to Eq. (5.10), a backup node's cost-effectiveness is defined as its average

backup resource consumption for each protected SFF. Since the backup resource consumption of each selected backup node is divided among the actual new unprotected SFFs that are in consideration, the overall backup resource consumption of all selected backup nodes equals $\sum_{k=1}^{d} cost(u_k)$. Therefore, by Lemma 4, we have Eq. (5.12), where d is the total number of SFFs in the PN.

$$\begin{aligned} \sum_{k=1}^{d} cost(u_k) &\leq \sum_{k=1}^{d} \frac{OPT}{(d-k+1)} \\ &\leq (1 + \frac{1}{2} + ... + \frac{1}{d}) * OPT = H_d * OPT \end{aligned} \tag{5.12}$$

□

5.5 Experimental Results

5.5.1 Simulation Settings

To evaluate the performances of the proposed schemes, we conduct our experiment in two network scenarios: (i) 24-node-43-link USNET [98], and (ii) random mesh networks. If not otherwise specified, in the network, each physical link is given a bandwidth randomly in the range of $[5, 20]$. The latency of each physical link is randomly set in the range of $[10, 30]$. The computing capacity is randomly set in the range of $[100, 300]$ for each physical node. In particular, each physical node randomly hosts four SFFs with four SFs, each of which randomly instantiates $[1, 4]$ SF instances. Note that multiple SF instances of the same SF may exist at the same physical node. The required computing resources for each SF instance are randomly set in a discrete-uniform range of $[10, 40]$. For each network scenario, we randomly generate 1000 runs to obtain the average results.

5.5.2 Evaluation Metrics

We evaluate the performance of the proposed algorithms in terms of the following metrics:

Backup resource consumption: Let *cost* represent the overall required backup resources for protecting all SFFs in the PN.

Resource saving ratio: Let S_β denote the proportion of computing resources saved through sharing to the total amount of computing resources that must be backed up.

Protected ratio: The selected backup nodes may end up not being able to protect all SF instances in the PN due to the lack of available resources, such as computing or bandwidth/latency on the physical links. Let R_β denote the protected ratio, which is the ratio between the number of protected SFF and the number of all SFFs.

To evaluate the sharing efficiency, we implement the internal sharing only (ISO) and external sharing only (ESO) algorithms by replacing Eq. (5.4) with Eq. (5.13) and Eq. (5.14), respectively.

$$c_m^{\text{internal}} = \sum_{n \in N} (\max_{\delta_{i,n} \subseteq E_n} r_{i,n} * x_m^{i,n}) \tag{5.13}$$

$$c_m^{\text{external}} = \sum_{\delta_{i,n} \subseteq E_n} (\max_{n \in N} r_{i,n} * x_m^{i,n}) \tag{5.14}$$

To evaluate the performance of the proposed FCRO algorithm, we also implemented a heuristic algorithm called the instance separated backup (ISB) algorithm that is extended from the existing work [99]. In ISB, the required computing resources of the SF instances in $\delta_{i,n}$ are allocated to different potential backup nodes in $\mathbb{P}(\delta_{i,n})$ separately, and each potential

backup node will protect at least one SF instance in $\delta_{i,n}$. For each $\delta_{i,n}$ in $\mathbb{U}$, the ISB algorithm repeats the following steps until $\mathbb{U}$ is empty: (i) identifies $\mathbb{P}_{i,n}$ for $\delta_{i,n}$; (ii) assigns the SF instance that has the largest required computing resources in $\delta_{i,n}$ to a potential backup node that has the largest available computing resources; (iii) repeats the last step until all SF instances in $\delta_{i,n}$ are protected separately; (iv) Adds the potential backup nodes into $\mathbb{B}$, removes $\delta_{i,n}$ from $\mathbb{U}$, and updates the backup resource consumption of $\mathbb{B}$.

5.5.3 Performance Evaluation in 24-node-43-link USNET

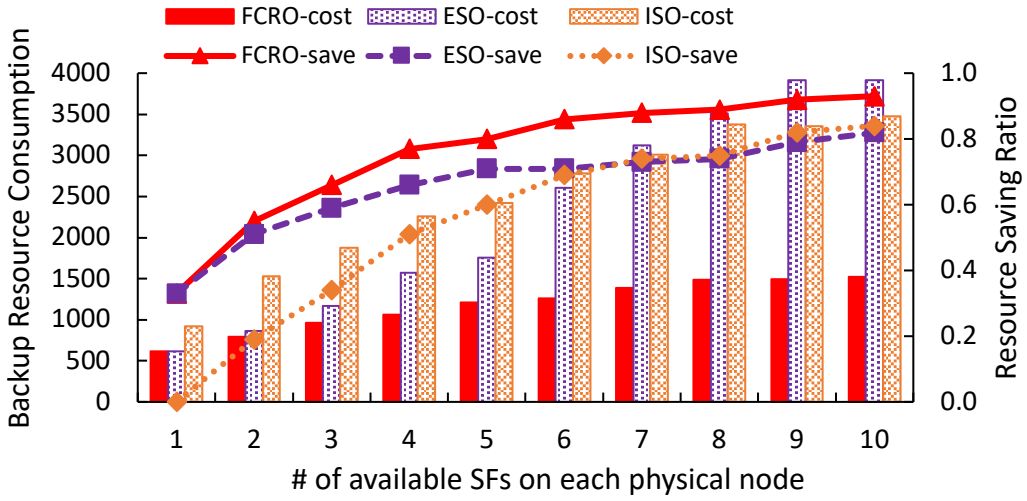


Figure 5.6: Impact of sharing efficiency.

To demonstrate the efficiency of the sharing, we set each physical node to host four SFFs with sufficient computing resources. Fig. 5.6 shows the performance of the proposed FCRO, ISO, and ESO regarding the backup resource consumption and the resource saving ratio when increasing the number of SFs on each physical node. The red solid bar (solid curve), purple stripes bar (dashed curve), and orange dotted bar (dotted curve) represent the performance of FCRO, ISO, and ESO, respectively. The left y-axis measures the bars for

the backup resource consumption, while the right y-axis measures the curves for the resource saving ratio.

In Fig. 5.6, as the number of SFs on each physical node increases, so does the number of SFFs; therefore, more running SF instances from the SFFs must be backed up. As a result, the three sharing protection schemes necessitate more backup resources to protect against the SFF failure. In comparison to ESO and ISO, FCRO requires less backup resources to back up the same number of SFs. This is due to the fact that FCRO takes advantages of both SFF internal and external sharing opportunities according to Eq. (5.10).

In Fig. 5.6, when each physical node has only one SF, the resource saving ratio of ISO is 0. This is because a physical node only has one SFF with SF instances running, and the SFF internal sharing cannot be applied at all. Due to the same reason, FCRO and ESO achieve the same performance when there is only one SF per physical node. Furthermore, ESO outperforms ISO when the number of SFs on each physical node is less than 7. When the number of SFs on each physical node is greater than 7, ISO, on the other hand, slightly outperforms ESO. This can be explained as follows. With more SFs within each physical node, the more identical SFFs can be found, resulting in more opportunities to share inside each physical node. As a result, the more identical SFFs within the physical node to be backed up, the more backup resources can be saved through the SFF internal sharing.

Fig. 5.7 shows the performance of FCRO and ISB when increasing the PN size. In Fig.5.7, each physical node has limited available computing resources. We increase the number of physical nodes and the number of physical links to maintain the average degree of nodes in

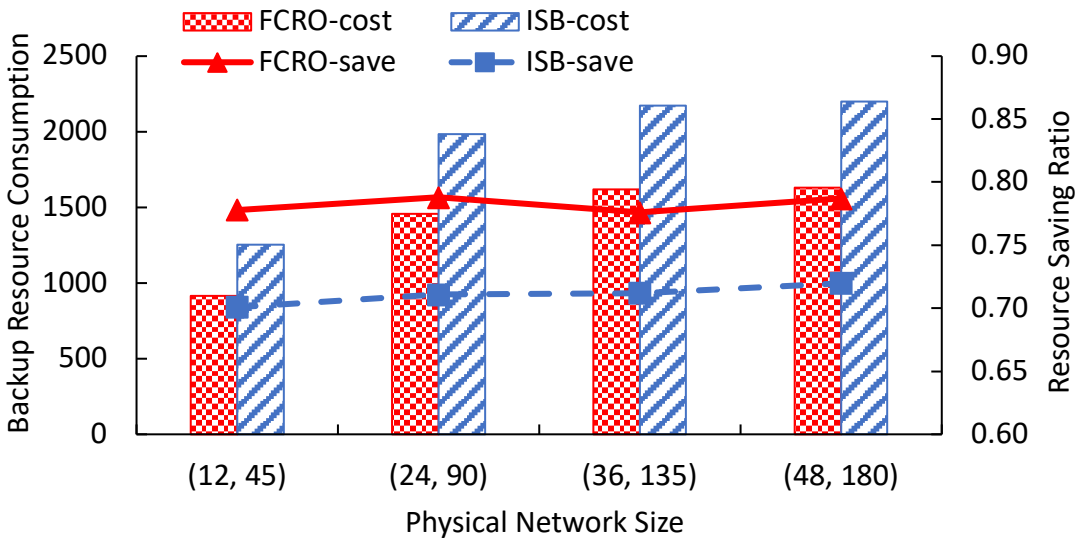


Figure 5.7: Impact of PN size.

the physical network. The red checker bar (solid curve) and blue stripes bar (dashed curve) represent the performance of FCRO, ISB, respectively. The left y-axis measures the bars for the backup resource consumption, while the right y-axis measures the curves for the resource saving ratio.

When increasing the PN size, more SFF connects to more running SF instances in the PN; thereby, both algorithms require more computing resources to protect the SFF failure. FCRO outperforms ISB in terms of the backup resource consumption and the resource saving ratio.

5.5.4 Performance Evaluation in Random Mesh Networks

Similar to [100], the random network is generated by constructing a tree with 40 nodes and then adding links to ensure the existence of 180 links in the network.

Figs. 5.8 and 5.9 present the performance of FCRO and ISB in the 24-node-43-link

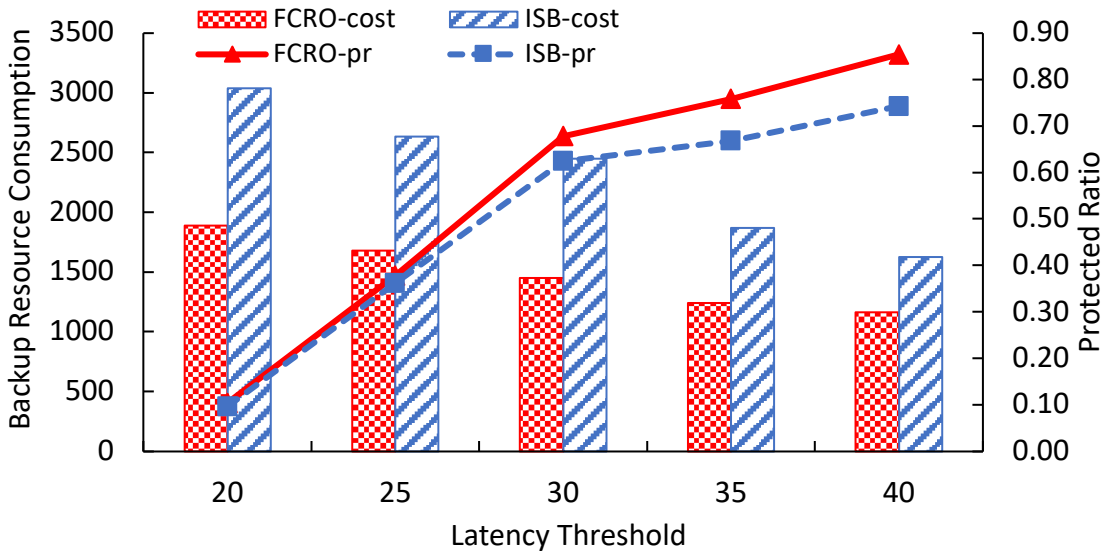


Figure 5.8: Latency threshold in USNET.

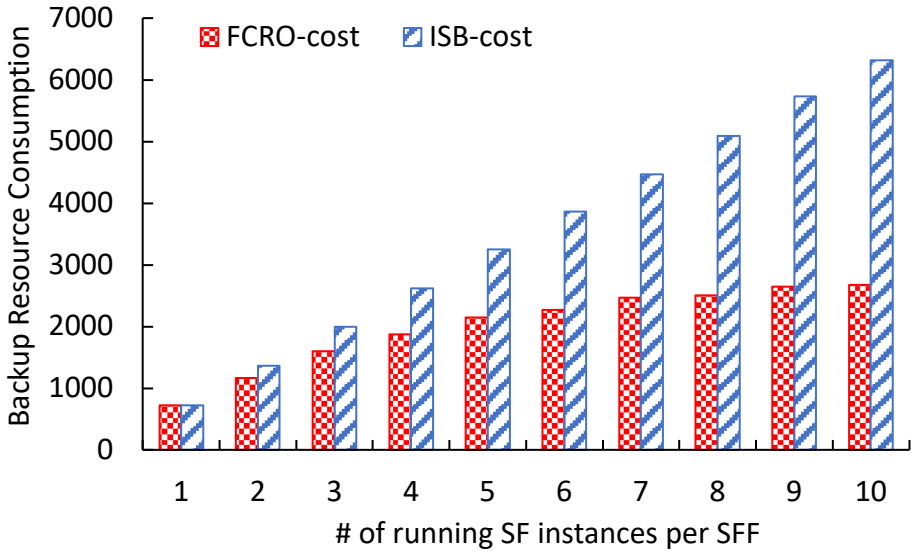


Figure 5.9: Number of running SF instances per SFF in USNET.

USNET. The backup resource consumption and the backup nodes number of FCRO and ISB are denoted by the red checker, and blue stripes bars, respectively. The resource saving ratio and the protected ratio of the FCRO and ISB are represented by the red solid, and blue dashed curves, respectively. In each sub-figure, the bars are measured by the left y-axis,

while the right y-axis measures the curves.

In Fig. 5.8, when the latency threshold of the backup path is increased, the backup resource consumption of both algorithms decreases while the protected ratio of both algorithms increases. The reasons are as follows. The relaxation of the latency threshold implies that the backup node can be located further away from the primary SF instances. As a result, more SFF internal and external sharing opportunities present.

Fig. 5.9 depicts how the number of running SF instances per SFF impacts the performance of FCRO and ISB when each physical node hosts four SFFs. When the number of running SF instances per SFF is increased, both algorithms require more backup resource consumption for backup. The backup resource consumption of FCRO is lower than ISB. This demonstrates the effectiveness of the AFB technique from FCRO. Since the AFB technique assigned the SF instances of these SFFs separately, it increases the sharing efficiency of the same amount of the backup resources from the backup nodes. As a result, even when the number of running SF instances per SFF is large, the overall backup resource consumption does not grow dramatically. In addition, we notice that when each SFF hosts one running SF instance, the performance of FCRO and ISB are similar. This is because FCRO has no chance to utilize the SFF internal sharing opportunity while ISB has no chance to separate the SF instance from the SFFs to different backup nodes.

Figs. 5.10 and 5.11 present the performance of FCRO and ISB in the 40-node-180-link random mesh network. Similarly, FCRO outperforms ISB in terms of the backup resource consumption and the protected ratio. It is worth noting that all schemes in this mesh

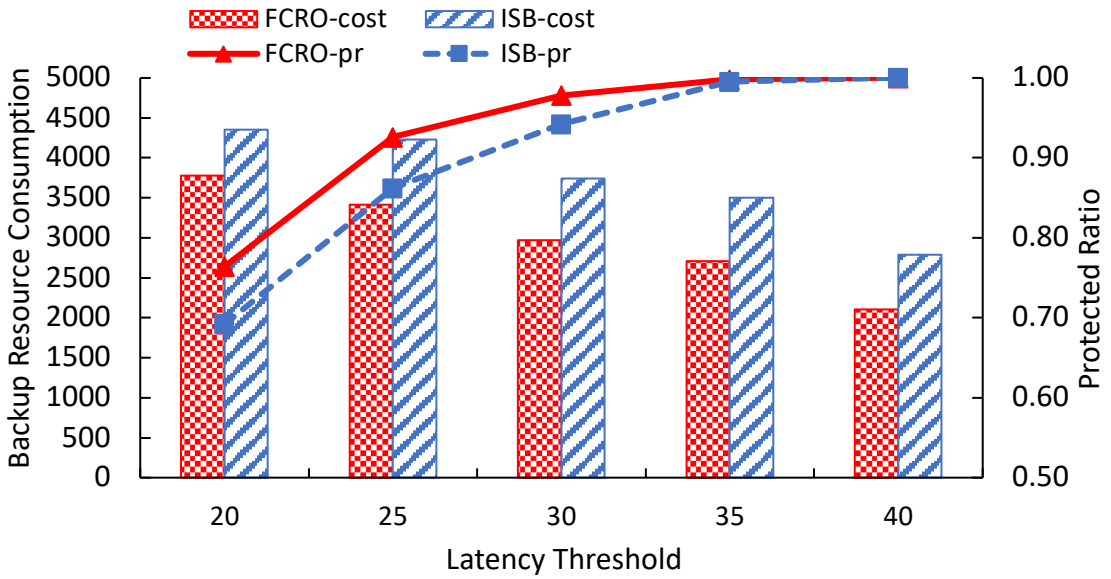


Figure 5.10: Latency threshold in random mesh networks.

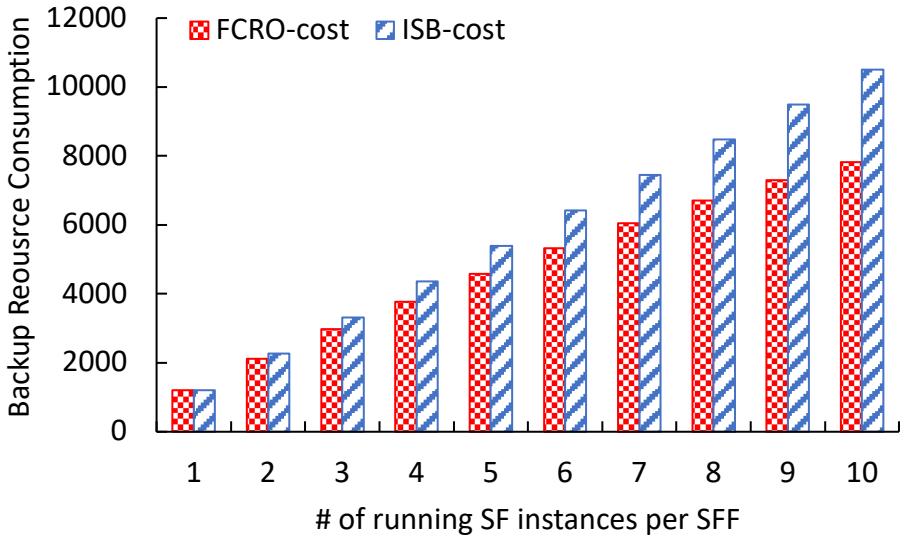


Figure 5.11: Number of running SF instances per SFF in random mesh networks.

network consume more backup resources, but are more reliable than in the 24-node-43-link USNET. This is due to the higher connectives in the 40-node-180-link network lead to more potential backup nodes and sharing. In our experiments, the average protected ratio for all schemes is 52.6% in the 24-node-43-link USNET and 91.6% in the 40-node-180-link network,

respectively. FCRO, on average, outperforms ISB by 31.6%.

5.6 Summary

In this work, we have studied how to provide SFF-based off-site sharing protection for all SF instances in a latency-constrained PN. We have identified the unique SFF internal and external sharing opportunities when providing the off-site protection against SFF failures in NFV. We have mathematically formulated a new problem, SFF-driven multi-instances failures and protection (SMFP), to minimize backup resource consumption. To optimize this problem, we have proposed an efficient heuristic algorithm called SFF-centralized resource optimization (FCRO), which builds upon three techniques: the backup auxiliary transferring, backup cost-effectiveness selection, and adaptive fit backup techniques. Our extensive simulation results have shown the efficiency of the proposed schemes, which significantly outperforms the approach directly extended from the existing work.

CHAPTER 6
SERVICE FUNCTION CHAIN FAILURE AND PROTECTION

6.1 Motivation

At the network edge, services are traditionally delivered through proprietary middleboxes [67]. Deploying and maintaining these middleboxes incurs high capital expenditures (CAPEX) and operational expenses (OPEX) [69]. To reduce the CAPEX and OPEX, network function virtualization (NFV) is introduced [101]. In NFV, traditional middleboxes are replaced by software-based service functions (SFs) [21]. A network service request (NSR) consists of multiple SFs along with corresponding resource demands (e.g., CPU, bandwidth) [68]. To deploy an NSR, the service provider can concatenate the required SFs to a service function chain (SFC) and embed it onto a shared physical network (PN) [80]. When an SF is embedded onto a physical node in the PN, an SF instance (or virtual machine) that runs the specific SF will be created. Once all the SFs are successfully embedded onto corresponding physical nodes, they will be connected to form a service function path (SFP) in the PN. The process of chaining and embedding an SFC to form a proper SFP is referred to as service function chaining and embedding (SFCE) [43].

Many SFC service requests require ultra-reliable performance for emerging 5G applications, such as physiological sensors for wearable medical devices or ultra-precision machining equipment in smart factories [96]. Such applications can fail and may result in severe financial losses or human casualties when there are no effective protection strategies [90]. Hence, it is imperative to provide fault-tolerance mechanisms to protect the SFC service.

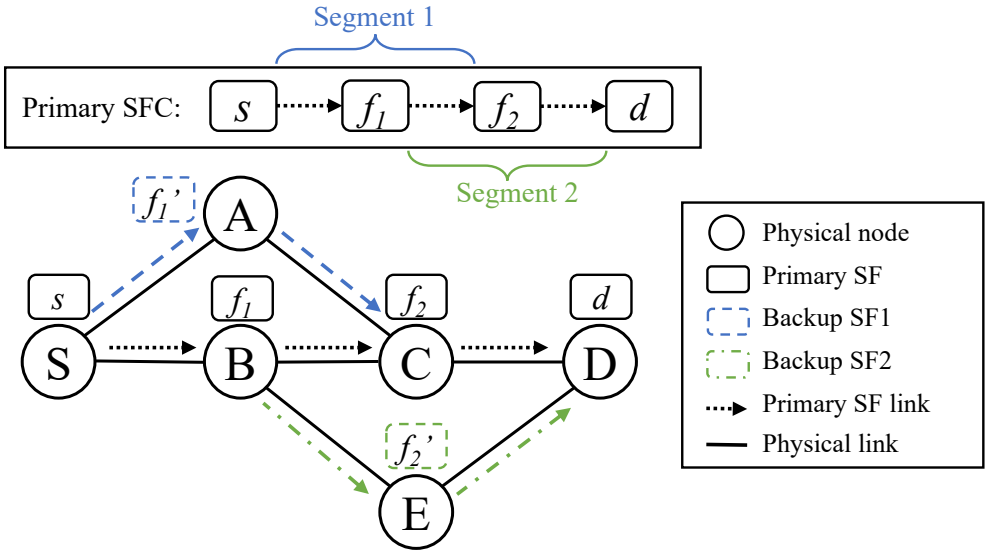


Figure 6.1: Example of chain segment protection.

The authors in [102] proposed a fast remapping algorithm to realize the rapid migration and response of an SFC service request when the SFC failed. The authors in [103] investigated a path recovery algorithm in SDN-enabled system to handle a single link failure. The authors in [48] explored a dedicated backup scheme to protect all nodes and links in a virtual network request by using minimum resource provisioning cost. The author in [104] proposed an efficient primary and backup SFC embedding algorithm to identify a node-link-disjoint backup path for the primary SFP in fog networks. However, such a node-link-disjoint backup path for the entire SFC may not even exist due to the PN topology, limited CPU or limited bandwidth resources. In Fig. 6.1, we assume that a primary SFC $s \dashrightarrow f_1 \dashrightarrow f_2 \dashrightarrow d$, is already embedded onto the primary SFP $S \rightarrow B \rightarrow C \rightarrow D$, as shown by the black dotted flow. To protect any failures along the primary SFC, we need embed a backup SFC $s \dashrightarrow f_1' \dashrightarrow f_2' \dashrightarrow d$ onto a backup SFP that is node-link-disjoint from the primary SFP.

As one can see, we cannot find a node-link-disjoint backup SFP to embed the backup SFC in Fig. 6.1. However, if we partition the backup SFC into chain segments, each chain-segment protects one part of the SFC. When any failure occurs, only the affected segment will be switched to its backup, while the other segments will remain undisturbed along the primary SFP. In Fig. 6.1, the primary SFC can be divided into two segments as f_1, and f_2. Here, we assume that only one failure occurs in the system [103]. We use f_i' to denote the backup SF of the primary SF f_i. When the first segment fails, a blue dashed backup path $S \rightarrow A(f_1') \rightarrow C$ can be employed, while the second segment remains on the path of $C(f_2) \rightarrow D$. Similarly, when the second segment fails, a green dot-dashed backup path $B \rightarrow E(f_3') \rightarrow D$ can be employed, and the first segment will stay on the path of $S \rightarrow B(f_1)$. It can be seen that the backup chain segment typically contains fewer SFs and is easier to be embedded when comparing the entire backup SFC. Identifying proper backup chain segments for effective protection is challenging, and different from the traditional segment path protection [105]. This is because the optimization process must jointly consider chain segmentation, SF node mapping, and SF link mapping.

In this chapter, for the first time, we investigate the problem of how to efficiently provide chain segment protection while minimizing the backup bandwidth consumption. We define a new problem named SFC failure with chain segment protection (SFCF-CSP) and prove its NP-hardness. To optimize SFCF-CSP, we propose an efficient heuristic algorithm called backup concatenation embedding and protecting (BCEP). Our analysis and simulation results show that the BCEP outperforms schemes that are directly extended from the existing

work.

The rest of this chapter is structured as follows. In Section 6.2, the chain segment protection is introduced. Section 6.3 mathematically formulates the SFCF-CSP problem. Section 6.4 describes the BCEP algorithm, and the simulation results are discussed in Section 6.5. Section 6.6 concludes our work.

6.2 Chain Segment Protection

6.2.1 Chain Segment Concatenation

Let $SFC = (s(f_0), f_1, ..., f_\tau, d(f_{\tau+1}))$ represent an SFC that contains a set of required SFs f_i, where $s(f_0)$ and $d(f_{\tau+1})$ are source and destination, and τ is the number of the SFs. Depending on whether or not SFs within an SFC are capable of changing their order of execution, the SFC can be classified as one of three categories: (1) SFC with no dependency, (2) SFC with partial dependency, and (3) SFC with strong dependency [106]. Without loss of generality, we assume that there is strong dependency among the SFs. That is, the SFs in an SFC have to follow a given order.

We define a chain segment, denoted by CS, includes a subset of required SFs of an SFC, which does not violate the dependency requirements in the original SFC. The SFC may have multiple possible CS followed by the strong dependency. Let $\mathbb{P}$ denote a set that consists of all possible chain segments of an SFC with the strong dependency requirements. Fig. 6.2 shows the $\mathbb{P}$ of a primary SFC $s \dashrightarrow f_1 \dashrightarrow f_2 \dashrightarrow f_3 \dashrightarrow d$, which includes six chain segments: $CS_1(f_1)$, $CS_2(f_2)$, $CS_3(f_3)$, $CS_4(f_1 \dashrightarrow f_2)$, $CS_5(f_2 \dashrightarrow f_3)$, and $CS_6(f_1 \dashrightarrow f_2 \dashrightarrow f_3)$.

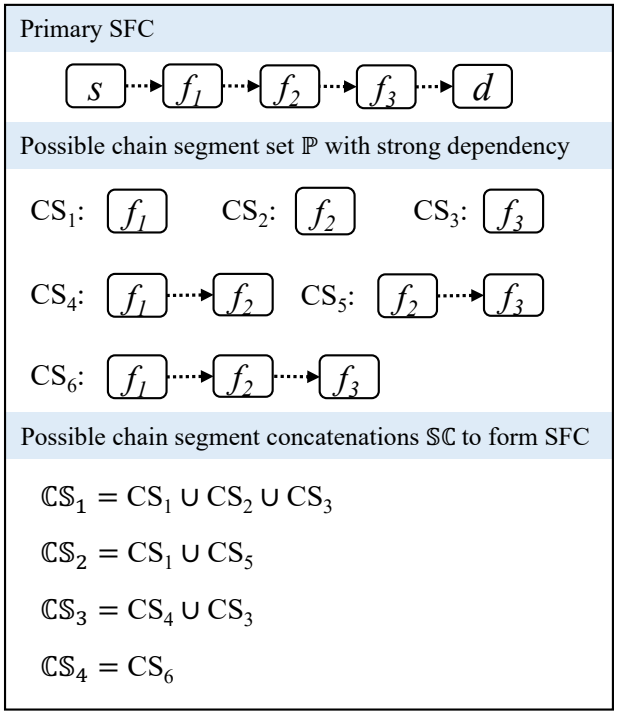


Figure 6.2: Example of a primary SFC with its $\mathbb{P}$ and $\mathbb{CS}$.

The concatenation of some chain segments (e.g., $CS_i \in \mathbb{P}$) can form the SFC: $SFC = CS_1 \cup ... \cup CS_k$. Note that, not all CSs in $\mathbb{P}$ are needed to form the SFC. A valid chain segment concatenation, denoted by $\mathbb{CS}$, must satisfy following constraints: (1) each SF in the SFC must belong to only one CS_i; and (2) the dependency relationships among the SFs are maintained. As shown in Fig. 6.2, there are four possible $\mathbb{CS}$ to form the primary SFC: $\mathbb{CS}_1 = CS_1 \cup CS_2 \cup CS_3$, $\mathbb{CS}_2 = CS_1 \cup CS_5$, $\mathbb{CS}_3 = CS_4 \cup CS_3$, and $\mathbb{CS}_4 = CS_6$.

6.2.2 Backup Concatenation

Traditionally, when any failures occur in the primary SFC, the traffic is rerouted to a disjoint backup SFC, in which all SFs in the primary SFC must be included [48]. However, such a total disjoint backup SFC may not be identified to provide protection. When the

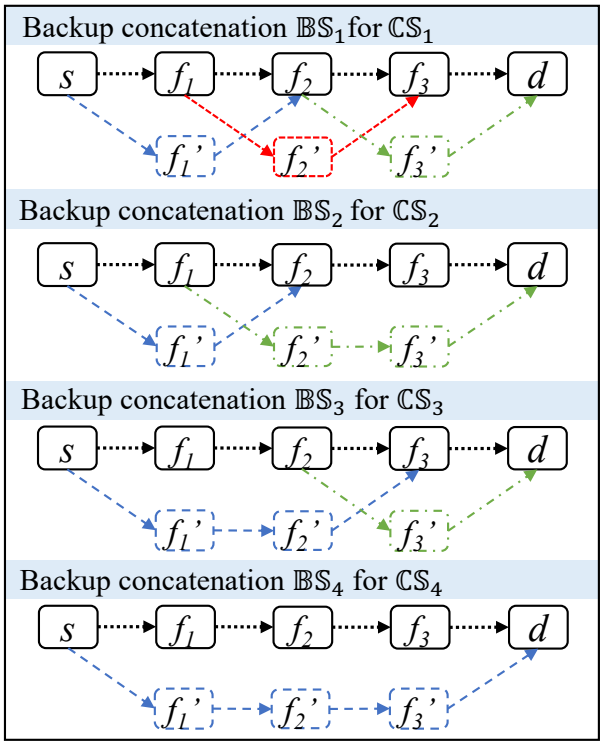


Figure 6.3: Backup concatenation $\mathbb{BS}$ for $\mathbb{CS}$.

primary SFC can be represented by a valid $\mathbb{CS}$, the backup of $\mathbb{CS}$, denoted by $\mathbb{BS}$, can provide equivalent protection. In specific, each chain segment CS_i in $\mathbb{CS}$ will be protected by a corresponding backup chain segment, denoted by BS_i in $\mathbb{BS}$ at a time. A valid $\mathbb{BS}$ must satisfy the following constraints: (1) each BS_i in $\mathbb{BS}$ must provide exactly same backup SFs for its corresponding primary CS_i in $\mathbb{CS}$; and (2) each BS_i in $\mathbb{BS}$ must maintain the same dependency relationship as CS_i in $\mathbb{CS}$.

Fig. 6.3 shows four backup concatenations $\mathbb{BS}$ for $\mathbb{CS}$ in Fig. 6.2. For example, the primary SFC can be concatenated by $\mathbb{CS}_1$, so it is protected by $\mathbb{BS}_1$, which consists of the blue BS_1 (f_1'), red BS_2 (f_2'), and green BS_3 (f_3'). Similarly, the primary SFC can be protected by $\mathbb{BS}_2$, which consists of the blue BS_1 (f_1'), and the green BS_5 ($f_2' \dashrightarrow f_3'$). The primary SFC

can also be protected by $\mathbb{BS}_3$, which consists of the blue BS_4 ($f'_1 \dashrightarrow f'_2$), and the green BS_3 (f'_3). Lastly, the primary SFC can be directly protected by $\mathbb{BS}_4$, which only consists of the blue BS_6($f'_1 \dashrightarrow f'_2 \dashrightarrow f'_3$). As one can see, $\mathbb{BS}_1$ can be treated as the traditional SF protection [78] by providing backup for each SF at a time, while $\mathbb{BS}_4$ can be seen as the traditional SFC protection.

In chain segment protection, the primary SFC and its backup concatenations must be embedded onto the PN in a disjoint manner. In this work, we assume that a physical node comprises both routing and computing components. A physical node failure is caused by its computing component while the routing component remains operational. Therefore, to ensure that the primary SFC and its backup concatenations are failure-disjointly accommodated onto the PN, they must satisfy following disjointness constraints: (1) every backup SF must be instantiated at a different physical node from any primary SF; and (2) every virtual detour link must be embedded onto a link-disjoint path. In addition, identifying such embedding results can be challenging, because they are directly affected by the PN topology, available SF and available computing/bandwidth resource on each physical node. For example, in Fig. 6.4, given the primary SFC that are delivered through the primary SFP $S \rightarrow A(f_1) \rightarrow B(f_2) \rightarrow C(f_3) \rightarrow D$, it shows that $\mathbb{BS}_3$ can be a feasible embedding result to satisfy the computation and routing disjointness constraints. In particular, the blue BS_4 is embedded onto the backup SFP $S \rightarrow E(f'_1) \rightarrow F(f'_2) \rightarrow D$, while the green BS_3 is embedded onto the backup SFP $B \rightarrow G \rightarrow H(f'_3) \rightarrow D$.

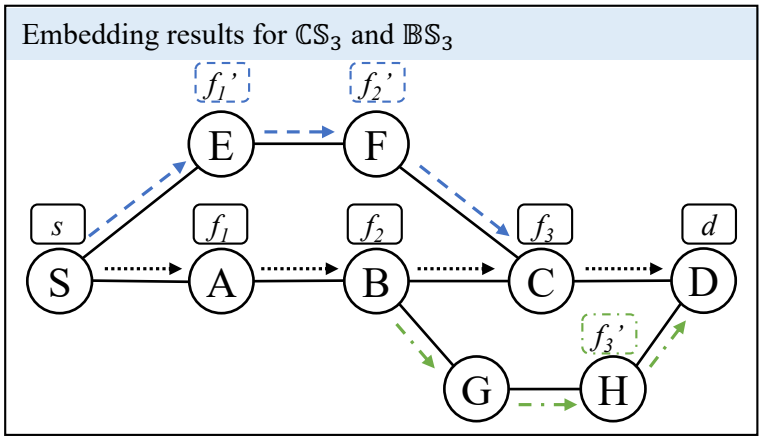


Figure 6.4: Embedding results for $\mathbb{CS}_3$ and $\mathbb{BS}_3$.

6.3 SFC Failure with Chain Segment Protection (SFCF-CSP)

6.3.1 Physical Network Model

The physical network (PN) is denoted by $PN = (N, L, F)$, where N, L and F represent the set of physical nodes, links and SF instances, respectively. Each physical node $n \in N$ can host a set of SF instances $F_n \in F$ and has a certain amount of available computing resources c_n. We use $l \in L$ to represent a physical link $l_{m,n} \in L$ for convenient, where $m, n \in N$ are its endpoints. Each l has a specific available bandwidth resources denoted by bw_l, and the unit cost of the bandwidth on l is $cost_l$. A physical path is composed of multiple physical links, and let P denotes a set of physical paths in the PN. As there might exist more than one path from one physical node to another one, we use $p_j \in P$ to represent the j-th path from Nodes m to n for convenient.

6.3.2 Network Service Request Model

A network service request (NSR) can be represented by a 2-tuple $NSR =< SFC, BW >$, where SFC denotes the required SFs, and BW is the bandwidth demands. In specific, for each SF $v \in SFC$, it has a primary SF instance $v_p \in \mathbb{CS}$ that is already embedded onto the physical node along the primary SFP. For each v_p, it has a corresponding backup SF instance $v_b \in \mathbb{BS}$, and demands a certain amount of computing resources c_{v_b}.

6.3.3 SFC Failure with Chain Segment Protection (SFCF-CSP)

The optimization problem of SFC failure with chain segment protection (SFCF-CSP) is defined as: given an PN and an NSR with embedded primary SF instances, how to protect the primary SFP with chain segment protection, such that (1) the following constraints are satisfied, (2) the backup bandwidth resource consumption is minimized. The objective function of the proposed SFCF-CSP is to minimize the backup bandwidth resource consumption as shown in Eq. (6.1). Table 6.1 shows the notations.

$$\min \quad \sum_{l \in L} \sum_{u_b \in \mathbb{BS}} \sum_{v_b \in \mathbb{BS}} z_l^{u_b, v_b} * BW * cost_l \tag{6.1}$$

Decision Variables: A binary variable $x_n^{v_b}$ is defined in Eq. (6.2) to determine whether the backup SF instance $v_b \in \mathbb{BS}$ is mapped onto $n \in N$. Eq. (6.3) defines a binary variable $y_n^{v_b}$ to determine whether Node $n \in N$ can provide $v_b \in \mathbb{BS}$. Eqs. (6.4) and (6.5) define the binary variables $z_l^{u_b, v_b}$ and $z_{p_j}^{u_b, v_b}$ to respectively indicate if the SF link from u_b to v_b is mapped onto l or p_j. A binary variable w_{SFC}^i is defined in Eq. (6.6) to determine whether BS_i is selected

Table 6.1: Notations.

Notation	Meaning
N	A set of physical nodes.
L	A set of physical links.
F	A set of SF instances.
m, n	A physical node in N.
F_n	A set of SF instances hosted by Node n.
c_n	Available computing resources on Node n.
$l_{m,n}$	A physical link between m and n.
bw_l	Available bandwidth resources on $l_{m,n}$.
$cost_l$	Cost of allocating unit bandwidth on $l_{m,n}$.
P	A set of physical path in the PN.
p_j	The j-th physical path between m and n.
v	An SF in SFC.
v_p	An embedded primary SF instance for v.
v_b	A backup SF instance for v.
c_{v_b}	Required computing resources of v_b.
$x_n^{v_b}$	A binary variable to determine whether the backup SF instance $v_b \in \mathbb{BS}$ is mapped onto $n \in N$.
$y_n^{v_b}$	A binary variable to determine whether $n \in N$ can provide v_b.
$z_l^{u_b,v_b}$	A binary variable to determine whether (u_b, v_b) is mapped onto l.
$z_{p_j}^{u_b,v_b}$	A binary variable to determine whether (u_b, v_b) is mapped onto p_j.
w_{SFC}^i	A binary variable to determine whether BS_i is selected to protect SFC.

to protect SFC.

$$x_n^{v_b} = \begin{cases} 1, & v_b \in \mathbb{BS} \text{ is mapped onto } n \in N. \\ 0, & \text{otherwise.} \end{cases} \tag{6.2}$$

$$y_n^{v_b} = \begin{cases} 1, & n \in N \text{ can provide } v_b \in \mathbb{BS}. \\ 0, & \text{otherwise.} \end{cases} \tag{6.3}$$

$$z_l^{u_b,v_b} = \begin{cases} 1, & \text{SF link from } u_b \text{ to } v_b \text{ is mapped onto } l. \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

$$z_{p_j}^{u_b,v_b} = \begin{cases} 1, & \text{SF link from } u_b \text{ to } v_b \text{ is mapped onto } p_j. \\ 0, & \text{otherwise.} \end{cases} \tag{6.5}$$

$$w_{SFC}^{i} = \begin{cases} 1, & BS_i \text{ is selected to protect } SFC. \\ 0, & \text{otherwise.} \end{cases} \tag{6.6}$$

SF Embedding Constraints: Eqs. (6.7) and (6.8) ensure that v_b is mapped onto exactly one physical node only if this physical node supports v_b. Eq. (6.9) ensures that n cannot host more than one backup SF instance from $\mathbb{BS}$. Eq. (6.10) constrains that the backup SF instances v_b has to be mapped on a disjoint physical node from the primary SF instance v_p. Eq. (6.11) ensures that n must provide enough available computing resources for v_b.

$$\sum_{n \in N} x_n^{v_b} = 1, \quad \forall v_b \in \mathbb{BS} \tag{6.7}$$

$$x_n^{v_b} \leq y_n^{v_b}, \quad \forall n \in N, \forall v_b \in \mathbb{BS} \tag{6.8}$$

$$\sum_{v_b \in \mathbb{BS}} x_n^{v_b} \leq 1, \quad \forall n \in N \tag{6.9}$$

$$x_n^{v_p} + x_n^{v_b} \leq 1, \quad \forall n \in N, v_p \in \mathbb{CS}, v_b \in \mathbb{BS} \tag{6.10}$$

$$x_n^{v_b} * c_{v_b} \leq c_n, \quad \forall n \in N, \forall v_b \in \mathbb{BS} \tag{6.11}$$

SF Routing Constraints: Eq. (6.12) ensures that each physical link hosts at most one virtual connection in $\mathbb{BS}$. This is the disjointness property to guarantee that the backup chain segments are mapped onto disjoint backup SFPs. Eq. (6.13) indicates the bandwidth resources constraints on each physical link.

$$\sum_{p_j \subset P} z_{p_j}^{u_b, v_b} \leq z_l^{u_b, v_b} \leq 1, \quad \forall l \in L, \forall u_b, v_b \in \mathbb{BS} \tag{6.12}$$

$$\sum_{u_b \in \mathbb{BS}} \sum_{v_b \in \mathbb{BS}} z_l^{u_b, v_b} * BW \leq bw_l, \quad \forall l \in L \tag{6.13}$$

Disjointness Constraints: Eqs. (6.14) and (6.15) ensure that the backup SF link from u_b to v_b is embedded onto node-link-disjoint paths with the embedded primary SF link from u_p to v_p. Eq. (6.16) constrains that every backup chain segment BS_i needs two virtual detour link connecting to the primary SFC.

$$\sum_{u_b \in \mathbb{BS}} \sum_{v_b \in \mathbb{BS}} z_l^{u_b, v_b} = 0, \quad if \quad z_l^{u_p, v_p} = 1, \tag{6.14}$$

$$\forall l \in L, \forall u_p, v_p \in \mathbb{CS}$$

$$z_l^{u_p,v_p} = 0, \quad if \quad \sum_{u_b \in \mathbb{BS}} \sum_{v_b \in \mathbb{BS}} z_l^{u_b,v_b} > 0, \tag{6.15}$$

$$\forall l \in L, \forall u_p, v_p \in \mathbb{CS}$$

$$\sum_{p_j \subset P} z_{p_j}^{u_p,v_b} = 2 * w_{SFC}^i, \tag{6.16}$$

$$\forall u_p \in \mathbb{CS}, \forall v_b \in \mathbb{BS}, \forall i \in [1, |\mathbb{BS}|]$$

Theorem 6.3.1. *The proposed SFCF-CSP problem is NP-hard.*

Proof. Several studies have shown that the dedicated SFC protection problem is NP-hard when network resources are limited [48, 104, 107, 108]. However, none of existing SFC protection methods can be used directly to address the SFCF-CSP problem. As described in Section II-B above, when an SFC is protected by a backup chain segment concatenation, the dedicated SFC protection can be considered as a special case of SFCF-CSP. This is because SFC can directly identify one BS to backup all the required SFs, such as $\mathbb{BS}_4$ in Fig. 6.3. As the special case of SFCF-CSP is NP-hard, SFCF-CSP is also NP-hard when network resources are limited. □

In the following section, we propose an efficient algorithm called backup concatenation embedding and protecting (BCEP) to optimize the proposed SFCF-CSP problem.

6.4 Backup Concatenation Embedding and Protecting (BCEP)

In this section, we first introduce the backup concatenation composing (BCC) technique. Based on the BCC technique, we propose an efficient backup concatenation embedding and protecting (BCEP) algorithm to minimize the backup bandwidth consumption.

6.4.1 Backup Concatenation Composing (BCC)

Intuitively, when any failures occur on the primary SFC, a failure-disjoint backup SFC is expected to be embedded onto the backup SFP. This backup SFP contains the least number of backup detours to protect the primary SFC, thereby minimizing the backup bandwidth consumption. However, such a disjoint backup SFP may not be found. To against any failures from the primary SFC, it must be partitioned and protected by employing chain segment protection.

In chain segment protection, when the primary SFC is given, each required SF must be protected by a chain segment, while the dependency requirements must be obeyed. Such proper chain segments can only be selected in the $\mathbb{P}$ of the primary SFC and form a valid chain segment concatenation $\mathbb{CS}$. To find these chain segments, the primary SFC must determine the appropriate breakpoints so that the corresponding backup concatenation can be composed. However, identifying the breakpoint can be challenging, because each SF may have multiple physical node candidates that can be embedded. In specific, the backup SF instance can only be embedded onto a physical node that is capable of hosting it as well as having enough computing resources. Thus, the identification of the most appropriate

breakpoint must be based on the information from a given PN.

In graph theory, betweenness centrality is a measure of centrality in a graph based on the shortest paths, which is often used to find nodes that serve as a bridge from one part of a graph to another [109]. Traditionally, the betweenness centrality of Node n can be calculated as shown in Eq. (6.17), where $p(n)$ and $p(S, D)$ represent the number of the shortest paths passing Node n and the total number of the shortest paths from source S to destination D, respectively.

$$BTC_n = \frac{p(n)}{p(S, D)} \tag{6.17}$$

The more shortest paths passing Node n, the more likely the SF embedded onto Node n will be connected to other SFs via the shortest path rather than going a longer detour. Accordingly, Node n with the higher BTC value can help to reduce the cost of constructing a backup SFP by connecting to other physical nodes, which provide the required backup SF instances via the shortest path. Hence, Node n can be a potential breakpoint for partition.

In the PN, each SF has several physical node candidates for embedding, and the backup SF can only be embedded onto these physical node candidates. We define a CBC factor as shown in Eq. (6.18) to set each SF a weight based on its physical node candidates in the PN. We specify a physical node candidate to host the SF v as n_v. The CBC factor is the sum of BTC values of n_v in the PN to measure the importance of the SF in the given primary SFC. The larger the CBC value, the more flexible v can be allocated. In other words, the more

likely a primary SFC will be partitioned into two chain segments via v.

$$CBC_v = \sum_{n_v \in N} \frac{p(n_v)}{p(S, D)} \tag{6.18}$$

With the CBC factor, we propose the backup concatenation composing (BCC) technique in Algorithm 8 to compose a valid backup chain segment concatenation $\mathbb{BS}$. In the BCC, Line 3 initializes a breakpoint SF v_x as empty. Line 4 calculates the CBC value for each SF v in SFC based on Eq. (6.18), and sorts all SFs in SFC in an ascending order. Line 5 identifies the SF v which has the largest CBC value in SFC as v_x, and partitions SFC into two chain segments as CS_1, and CS_2 via v_x. Line 6 constructs the corresponding BS_1 and BS_2, and generates backup detour links for each BS to connect with SFC. Line 7 updates $\mathbb{BS}$, and lastly, Line 6 returns this $\mathbb{BS}$.

Algorithm 8 Backup Concatenation Composing (BCC)

1: **Input:** PN, SFC;
2: **Output:** $\mathbb{BS}$;
3: Initialize a breakpoint SF v_x as $\emptyset$;
4: Calculate the CBC value for each SF v in SFC based on Eq. (6.18), and sort v in SFC in an ascending order;
5: Identify v which has the largest CBC value in SFC as v_x, and partition SFC into CS_1 and CS_2 via v_x;
6: Construct the corresponding BS_1 and BS_2, and generate backup detour links to connect with the primary SFC;
7: Update $\mathbb{BS} = BS_1 \cup BS_2$;
8: **return** $\mathbb{BS}$.

6.4.2 Backup Concatenation Embedding and Protecting (BCEP)

In chain segment protection, the backup SFC should have as few partitions as possible to minimize backup bandwidth consumption. Once a backup SFC creates more partitions,

it has more backup chain segments. Since each backup chain segment requires two backup detour links for connecting to the primary SFC, more partitions imply a higher consumption of backup bandwidth. Therefore, the protection of the primary SFC should start with the 0 partition, that is, prioritizing to determine whether the entire backup SFC can be embedded on the PN.

By applying the BCC technique, we propose the backup concatenation embedding and protecting (BCEP) in Algorithm 9. Let $SFP_{\mathbb{BS}}$ represent a set of backup paths for $\mathbb{BS}$, $cost_{\mathbb{BS}}$ denote the backup bandwidth consumption for $\mathbb{BS}$, and $cost_{BS_i}$ denote the backup bandwidth consumption for BS_i. In BCEP, Line 3 first initializes $cost_{\mathbb{BS}} = 0$, and $cost_{BS_i} = 0$. Line 4 creates three sets: $\mathbb{BS}$ and $SFP_{\mathbb{BS}}$ as two empty sets, and $\mathbb{U} = SFC$. Line 5 identifies the given primary SFP for SFC. Since each physical link has a certain bandwidth cost, the bandwidth-aware shortest SFP can be obtained by applying Dijkstra's algorithm beforehand. Line 6 deletes all physical nodes and links of SFP from PN. As mentioned earlier, the primary SFC should have as few partitions as possible. In BCEP, we will start with the SFC in partition 0, first identifying a failure-disjoint backup SFC. To do so, Line 7 creates a backup SFC BS_0 for SFC and obtains the shortest failure-disjoint backup path SFP_{BS_0}. If such SFP_{BS_0} can be identified in Line 8, Line 9-12 removes SFC from $\mathbb{U}$, adds BS_0 into $\mathbb{BS}$, adds SFP_{BS_0} into $SFP_{\mathbb{BS}}$, and updates $cost_{\mathbb{BS}} = cost_{BS_0}$. Otherwise, the BCEP executes Line 13-25, whereas the chain segment protection is activated. In specific, while $\mathbb{U}$ is not empty, Line 15 calls BCC(PN, $\mathbb{U}$), and Line 16 obtains the shortest failure-disjoint backup for each BS_i in $\mathbb{U}$. If BCEP identifies SFP_{BS_i} in Line 17, Line 18-20 moves CS_i

from $\mathbb{U}$, adds SFP_{BS_i} into $SFP_{\mathbb{BS}}$, and updates $cost_{\mathbb{BS}} = cost_{\mathbb{BS}} + cost_{BS_i}$. Otherwise, BCEP updates $\mathbb{BS}$ by removing the invalid BS_i which does not have the embedding results. BCEP repeats the above Line 14-24 until $\mathbb{U}$ is empty. Lastly, Line 26 returns the embedding results.

Algorithm 9 Backup concatenation embedding and protecting (BCEP)

1: **Input:** PN, SFC, SFP;
2: **Output:** $SFP_{\mathbb{BS}}$, $cost_{\mathbb{BS}}$;
3: Initialize $cost_{\mathbb{BS}} = 0$, and $cost_{BS_i} = 0$;
4: Create three sets $\mathbb{BS} = \emptyset$, $SFP_{\mathbb{BS}} = \emptyset$, $\mathbb{U} = SFC$;
5: Identify the given primary SFP for SFC;
6: Delete all physical nodes and links of SFP from PN;
7: Create a backup SFC BS_0 for SFC, and obtain the shortest failure-disjoint backup path SFP_{BS_0};
8: **if** $SFP_{BS_0} == true$ **then**
9: Remove SFC from $\mathbb{U}$
10: Add BS_0 into $\mathbb{BS}$
11: Add SFP_{BS_0} into $SFP_{\mathbb{BS}}$
12: Update $cost_{\mathbb{BS}} = cost_{BS_0}$;
13: **else**
14: **while** $\mathbb{U} \neq \emptyset$ **do**
15: Call BCC($PN, \mathbb{U}$);
16: Obtain the shortest failure-disjoint backup path for each BS_i in $\mathbb{BS}$;
17: **if** $SFP_{BS_i} == true$ **then**
18: Remove CS_i from $\mathbb{U}$
19: Add SFP_{BS_i} into $SFP_{\mathbb{BS}}$
20: Update $cost_{\mathbb{BS}} = cost_{\mathbb{BS}} + cost_{BS_i}$;
21: **else**
22: Update $\mathbb{BS}$ by removing the invalid BS_i;
23: **end if**
24: **end while**
25: **end if**
26: **return** $SFP_{\mathbb{BS}}$, $cost_{\mathbb{BS}}$.

In the BCEP algorithm, the process with the highest running time complexity is to calculate the CBC values to identify the most appropriate breakpoint SF to partition SFC. In the worst case, when every SF is the breakpoint SF for SFC, the time complexity of this

algorithm is $\mathcal{O}(\tau(|N||L| + |N|^2 log|N|))$, where τ is the number of the required SFs, $|N|$ is the number of physical nodes, and $|L|$ is the number of physical links.

Table 6.2: PN Information in Fig. 6.4.

Physical node	A	B	C	D	E	F	G	H	S
Available SF	f_1	f_2	f_3	d	f_1	f_2	f_2	f_3	s
Available CPU	6	7	5	/	6	8	3	7	/

We use the PN in Fig. 6.4 to illustrate how the BCEP algorithm works. Table 6.2 shows the PN information in Fig. 6.4, which includes each physical node's available SFs and available computing resources. Here, we assume that each SF requires 5 CPUs to install the backup SF instance, and the unit cost of the bandwidth is $cost_l = 1$. Given an SFC $s \dashrightarrow f_1 \dashrightarrow f_2 \dashrightarrow f_3 \dashrightarrow d$, which is delivered via the shortest primary SFP $S \rightarrow B \rightarrow C \rightarrow D$. To begin with, BCEP initializes $\mathbb{BS}$ and $SFP_{\mathbb{BS}}$ as empty, and $\mathbb{U} = \{f_1, f_2, f_3\}$. The BCEP algorithm removes Node S, B, C, and D from the PN. Next, BCEP creates a backup SFC $f_1' \dashrightarrow f_2' \dashrightarrow f_3' \dashrightarrow d$, and determines that there is no shortest backup path in the PN to embedded it. Following that, BCEP calls the BCC technique to find the first breakpoint SF f_2. Specifically, BCEP calculates the CBC values of all SFs and identifies that f_2 has the largest CBC value. The primary SFC is concatenated by $\mathbb{CS}$, which consists of $CS_1(f_1 \dashrightarrow f_2)$ and $CS_2(f_3)$. Next, BCEP constructs the corresponding $BS_1(f_1' \dashrightarrow f_2')$ and $BS_2(f_3')$ with backup detour links. The BCEP obtains the shortest failure-disjoint backup path for BS_1 and BS_2, respectively. Here, BS_1 is delivered via $S \rightarrow E(f_1') \rightarrow F(f_2')$ and its backup cost is 3, while BS_2 is delivered via $B \rightarrow G \rightarrow H(f_3') \rightarrow D$ and its backup cost is 4. Lastly, BCEP removes f_1, f_2 and f_3 from $\mathbb{U}$, adds $S \rightarrow E(f_1') \rightarrow F(f_2')$ and

$B \rightarrow G \rightarrow H(f_3') \rightarrow D$ into $SFP_{\mathbb{BS}}$, and updates $cost_{\mathbb{BS}} = 3 + 4 = 7$. Thus, the overall backup bandwidth consumption of the primary SFC is 7.

6.5 Experimental Results

6.5.1 Simulation Settings

To evaluate the performances of the proposed schemes, we conduct our experiments in a 40-node-180-link random network with a minimum node degree of 3. The parameters are generated as follows if not otherwise specified, similar to the simulation settings in [21, 78, 100]. Each physical node has enough computing resources to host multiple SF instances and provides [2, 5] types of SFs. Each physical link has bandwidth resources set in a range of [5, 20] and the cost of each link is set in a range of [3, 8]. In addition, the network provides 8 types of SFs, and each SF has several physical node candidates to be hosted in a range of [5, 8]. For each NSR, we set the number of required SFs in a range of [6, 24], the computing resources for instantiating the required SF are set in a range of [10, 20], and the bandwidth demands are set in a range of [1, 5]. Specifically, we randomly generate the source and destination of an NSR, and the physical node that hosts the primary SF instance is randomly selected from physical node candidates. We generate 1000 copies of 40-node-180-link random networks and 100 different NSRs.

6.5.2 Evaluation Metrics

Blocking probability: is used to measure the backup blocking ratio of each approach when deploying the corresponding backup path(s). An NSR's backup will be blocked when

(1) lacks backup computing resources to embed backup SF; (2) lacks backup bandwidth resources to identify failure-disjoint backup path(s).

Average backup cost: is used to measure the average backup bandwidth resources employed by each scheme for each NSR. A smaller average backup cost means that we need to assign fewer bandwidth resources. It also means fewer bandwidth resources are reserved along all the backup paths.

Average recovery time: is used to measure the average computation time per NSR employed by each scheme.

Furthermore, we extend and implement an existing protection method in [48] and name it protection for SFC only (P-SFCO). The NSR will be blocked immediately if the algorithm cannot find a proper backup path for the primary SFC. We also implement protection for chain segment only (P-CSO) method by directly applying Line 9-17 in the proposed BCEP algorithm. Specifically, P-CSO provides backup chain segments for the primary SFC, which always start with identifying the first breakpoint SF for partition. The NSR will be blocked immediately if the algorithm cannot find proper backup paths for each chain segment.

6.5.3 Performance Evaluation

Fig. 6.5 shows the blocking probability of three schemes regarding bandwidth capacity on each physical link. When increasing the bandwidth capacity, the blocking probability of BCEP and P-CSO is lower than that of P-SFCO. This is because, when bandwidth resources are insufficient, P-SFCO cannot find an entire backup SFC for each NSR and guarantee total-disjoint when the backup SFC is of the same length as the primary SFC. On the other hand,

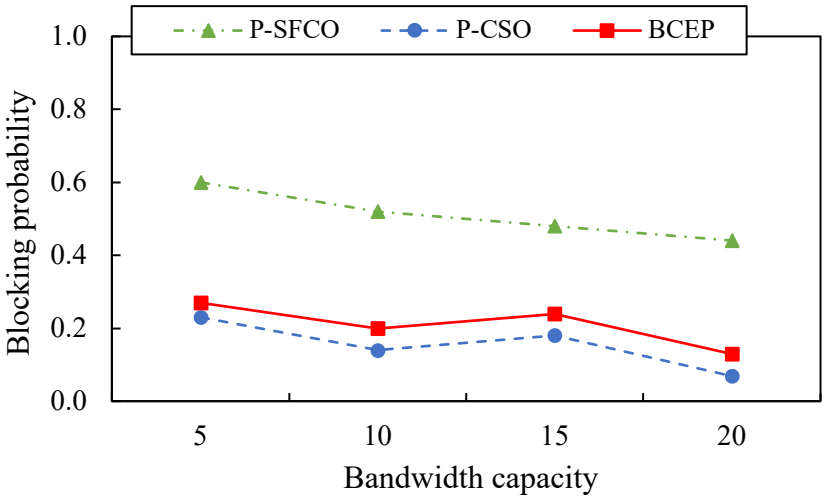


Figure 6.5: Blocking probability vs. bandwidth capacity.

BCEP and P-CSO provide chain segment protection for the primary SFC. The length of the chain segments is typically shorter, which results in fewer backup resources reserved for each chain segment than that of P-SFCO. P-CSO has a slightly lower blocking probability than BCEP because it directly starts with identifying the first breakpoint SF (1 partition) instead of identifying a failure-disjoint backup SFC (0 partition) before determining it, as BCEP does. Without loss of generality, for the same primary SFC, if it can find a failure-disjoint backup SFC, it must be able to find a backup chain segment concatenation, but not vice versa.

In Fig. 6.6, we can see that the blocking probability of three schemes increased while increasing the number of required SFs per NSR. The reason is that, as the length of the SFC becomes longer, the length of the entire backup SFC and backup chain segments also becomes longer. Thus, finding the corresponding backup path can be challenging. Since the length of the backup chain segment and the corresponding backup SFP is shorter than the

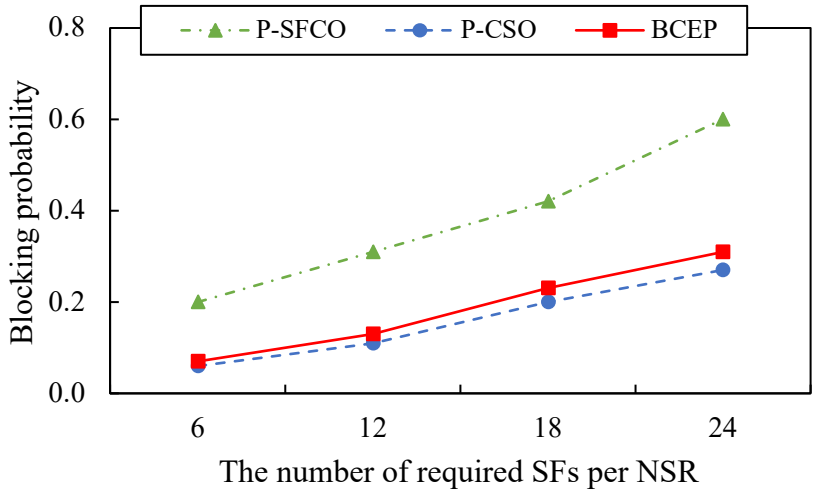


Figure 6.6: Blocking probability vs. the number of required SFs per NSR.

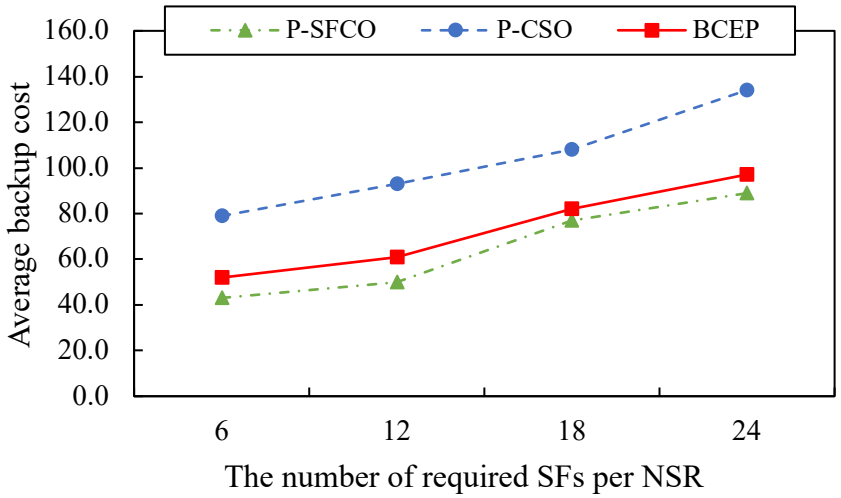


Figure 6.7: Average backup cost vs. the number of required SFs per NSR.

length of the primary SFC and SFP. When failures occur, the blocking probability of BCEP and P-CSO is lower than that of P-SFCO.

In Fig. 6.7, when increasing the number of required SFs per NSR, the P-SFCO outperforms the P-CSO and BCEP by using fewer reserved backup bandwidth resources. This can be explained by the following. The P-SFCO always identifies an entire backup SFC for pro-

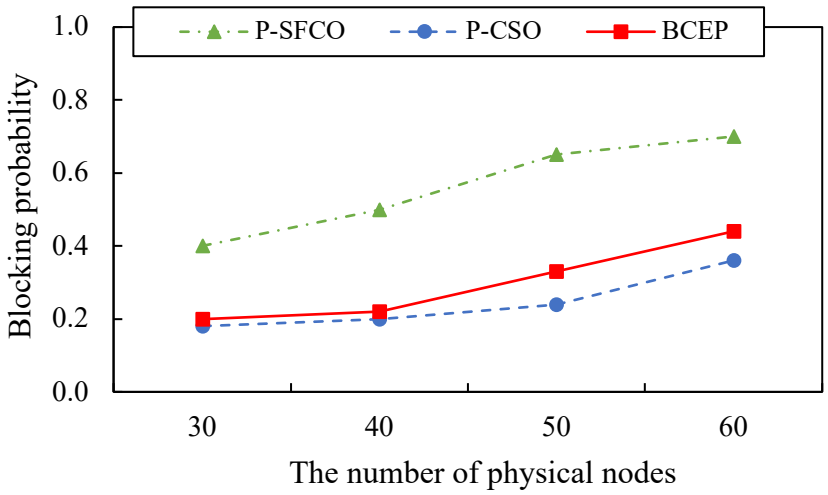


Figure 6.8: Blocking probability vs. the number of physical nodes.

tecting the primary SFC. Since its backup SFP contains the least number of backup detours, thereby it has the least backup bandwidth consumption. On the contrary, as more SFs are required in each NSR, the primary SFC may be partitioned more for P-CSO, which increases the number of backup chain segments of P-CSO. As a result, the number of backup detours used to connect to the primary SFP also increases, P-CSO has to provide more average backup cost compared to P-SFCO and BCEP.

In Fig. 6.8, when the number of physical nodes increases, the blocking probability of the three algorithms increases. This is because the physical links remain constant at 180. Thereby when the PN has only 30 physical nodes, the average degree per node is relatively high. Hence, there is a higher probability of finding the shortest backup path regardless of whether it is for a backup SFC or chain segment. As a result, P-CSO and BCEP's blocking probability can be reduced to almost 20%. However, as the number of physical nodes is increased, the node degree of each physical node decreases, causing the whole graph to

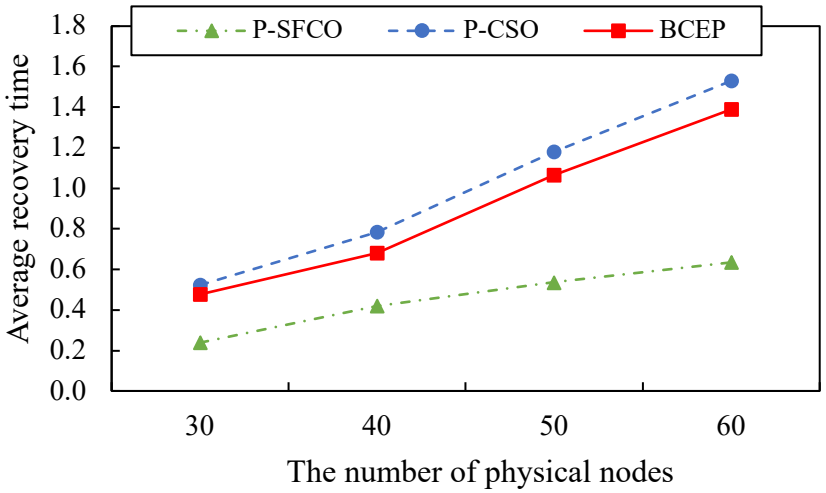


Figure 6.9: Average recovery time vs. the number of physical nodes.

become more sparse. Therefore, finding a proper backup path becomes more difficult among these three algorithms.

In Fig. 6.9, we can see that the average recovery time is longer for all three schemes when the number of physical nodes in the PN increases. P-SFCO performs a faster average recovery time than P-CSO and BCEP. Moreover, we note that the performance of P-CSO and BCEP is close. This can be explained by the following. If the number of physical nodes becomes larger, the computation of CBC values will consume more time to find a proper breakpoint SF and thus form a backup chain segment concatenation. Consequently, even if the required number of SFs in NSR remains the same, the computation of CBC values will take longer as the number of physical nodes increases.

To summarize, when the number of required SFs per NSR increases, the P-SFCO algorithm has the lowest average backup cost among the three algorithms, due to the least backup detours. However, it also has the highest blocking probability, which can provide limited sup-

port when bandwidth resources are limited. In contrast, the P-CSO algorithm outperforms the three algorithms in blocking probability and always has the lowest blocking probability. This highlights the advantages of our proposed chain segment protection. However, despite the lower blocking probability, we observe that the average backup cost of P-CSO is the highest when increasing of the number of required SFs per NSR, as the more chain segments are generated, the more the backup detour links are consumed. Overall, BCEP outperforms the other two schemes by minimizing the backup bandwidth consumption while keeping the blocking probability as low as possible.

6.6 Summary

In this chapter, we have investigated a new problem of SFC failure with chain segment protection (SFCF-CSP) for the first time. We have introduced chain segment protection to protect the primary SFC. After mathematically formulating the problem of SFCF-CSP and proving its NP-hardness, we have proposed an efficient algorithm called backup concatenation embedding and protecting (BCEP) based on the proposed backup concatenation composing (BCC) technique. Our extensive analysis and simulation results have shown that the proposed BCEP outperforms the algorithms directly extended from the existing works.

CHAPTER 7
DISSERTATION CONCLUSION

7.1 Dissertation Overview

While NFV provides more flexibility and efficiency for network service delivery, it also introduces new challenges to ensure network robustness, where network reliability with NFV has become a significant concern. In this dissertation, for the first time, we comprehensively study and propose a series of problems for efficiently delivering the network services to clients with backup resource allocation approaches against different network failures in NFV.

In Chapter 2, we studied the problem of protecting the running NFV services from SF instance failure in unique function networks. We named this problem the VNF failure protection cover (VFPC) while considering the latency, computing, and off-site backup constraints. We have formulated a mathematical model for the VFPC problem to minimize the number of VNF backup nodes and proposed an $ln(|N|) + 1$-approximation algorithm, namely, protection cover list-based VNF protection (PCL-VP), where $|N|$ represents the number of physical nodes in the SN. Our simulation results show that the proposed PCL-VP algorithm outperforms the minimum dominating set based on the VNF protection (MDS-VP) algorithm.

In Chapter 3, we have investigated the problem of protecting NFV services against SF instance failure in multiple functions networks. We call it the latency-bounded off-site virtual node protection (LOVNP) problem, which considers the SF computation, the QoS latency, and the off-site virtual node protection constraints. We have proved the NP-hardness of the

LOVNP problem and proposed an efficient logarithm-approximate algorithm called protection centrality-based pairwise node protection (PC-PNP). The proposed PC-PNP algorithm employs two novel techniques: pairwise node protection and protection centrality. The former ensures node pairs are efficiently identified to meet the demands of the off-site virtual node protection constraint, while the latter tries to maximize the backup sharing among the selected backup nodes. We have also proved that the proposed PC-PNP algorithm is logarithm-approximate. Our extensive simulation and analysis have shown that the proposed PC-PNP algorithm outperforms the algorithms that are extended from the existing work.

In Chapter 4, we have studied a shared backup resource allocation problem to defend against SFF failure. We defined a novel problem called the off-site type-oriented SFF failure protection (TSFP) in NFV and mathematically formulated this problem. We have proposed an efficient heuristic algorithm called type-oriented off-site shared protection optimization (TOPO) and proved its logarithm approximation. Extensive simulation results have verified that the proposed TOPO outperforms the approaches directly extended from the existing works.

In Chapter 5, we have studied how to provide SFF-based off-site sharing protection for all SF instances in a latency-constrained PN. We have identified the unique SFF internal and external sharing opportunities when providing off-site protection against SFF failures in NFV. We have mathematically formulated a new problem, SFF-driven multi-instances failures and protection (SMFP), to minimize backup resource consumption. To optimize this problem, we have proposed an efficient heuristic algorithm called SFF-centralized resource

optimization (FCRO), which builds upon three techniques: the backup auxiliary transferring, backup cost-effectiveness selection, and adaptive fit backup techniques. Our extensive simulation results have shown the efficiency of the proposed schemes, which significantly outperforms the approach directly extended from the existing work.

In Chapter 6, we have investigated a new problem of SFC failure with chain segment protection (SFCF-CSP) for the first time. We have introduced chain segment protection to protect the primary SFC. After mathematically formulating the problem of SFCF-CSP and proving its NP-hardness, we have proposed an efficient algorithm called backup concatenation embedding and protecting (BCEP) based on the proposed backup concatenation composing (BCC) technique. Our extensive analysis and simulation results have shown that the proposed BCEP outperforms the algorithms directly extended from the existing work.

7.2 Future Direction

While this dissertation investigated state-of-the-art problems in delivering NFV services with reliable resource allocation strategies, two future directions may be explored. First, this dissertation considers providing chain segment protection against SFC failure. By concatenating more backup chain segments, we can extend the proposed strategies in a shared manner, such that multiple primary chain segments from different SFCs can share the same backup chain segment for fault tolerance. Second, when SFC services are delivered, the existing approaches cannot simultaneously handle overlapping multiple failure types. As an example, the link and SF instances may fail simultaneously. It is then possible to extend the

proposed protection strategies to achieve k-fault tolerance for the SFC to keep a high level of reliability.

REFERENCES

[1] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang, "A survey of network virtualization techniques for internet of things using SDN and NFV," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 35:1–35:40, 2020.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS Operating System Review*, M. L. Scott and L. L. Peterson, Eds., 2003, pp. 164–177.

[3] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 9, no. 4, pp. 373–392, 2012.

[4] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, 2010.

[5] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Netw.*, vol. 167, p. 106984, 2020.

[6] Z. Shang, "Performance evaluation of the control plane in openflow networks," Ph.D. dissertation, Free University of Berlin, Dahlem, Germany, 2020.

[7] C. Ternon, J. Goossens, and J.-M. Dricot, "Ftt-openflow, on the way towards real-time sdn," *SIGBED Rev.*, vol. 13, no. 4, p. 49–54, 2016.

[8] "Etsi gs nfv-man 001: Network functions virtualisation (nfv); management and orches-

tration," *v. 1.1.1*, Dec. 2014.

[9] P. Quinn and T. Nadeau. Problem statement for service function chaining. [Online]. Available: https://tools.ietf.org/html/rfc7498

[10] M. Veeraraghavan, T. Sato, M. Buchanan, R. Rahimi, S. Okamoto, and N. Yamanaka, "Network function virtualization: A survey," *IEICE Trans. Commun.*, vol. 100-B, no. 11, pp. 1978–1991, 2017.

[11] D. Zheng, C. Peng, X. Liao, L. Tian, G. Luo, and X. Cao, "Towards latency optimization in hybrid service function chain composition and embedding," in *proc. IEEE INFOCOM*, 2020, pp. 1539–1548.

[12] N. Shahriar, S. R. Chowdhury, R. Ahmed, A. Khan, S. Fathi, R. Boutaba, J. Mitra, and L. Liu, "Virtual network survivability through joint spare capacity allocation and embedding," *IEEE J. Sel. Areas Commun*, vol. 36, no. 3, pp. 502–518, March 2018.

[13] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "A dynamic reliability-aware service placement for network function virtualization (NFV)," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 318–333, 2020.

[14] M. Wang, B. Cheng, and J. Chen, "Joint availability guarantee and resource optimization of virtual network function placement in data center networks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 821–834, 2020.

[15] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, "On the resiliency of virtual network functions," *IEEE Commun. Magaz.*, vol. 55, no. 7, pp. 152–157, 2017.

[16] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang, "A

survey of network virtualization techniques for internet of things using sdn and nfv," *ACM Comput. Surv.*, vol. 53, no. 2, 2020.

[17] D. Zheng, H. Gu, W. Wei, C. Peng, and X. Cao, "Network service chaining and embedding with provable bounds," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7140–7151, 2021.

[18] M. Zeng, W. Fang, J. J. P. C. Rodrigues, and Z. Zhu, "Orchestrating multicast-oriented nfv trees in inter-dc elastic optical networks," in *proc. IEEE ICC*, 2016.

[19] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[20] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *proc. IEEE INFOCOM*, 2018, pp. 2132–2140.

[21] D. Zheng, C. Peng, X. Liao, L. Tian, G. Luo, and X. Cao, "Towards latency optimization in hybrid service function chain composition and embedding," in *proc. IEEE INFOCOM*, 2020, pp. 1539–1548.

[22] D. Zheng, C. Peng, E. Guler, G. Luo, L. Tian, and X. Cao, "Hybrid Service Chain Deployment in Networks with Unique Function," in *proc. IEEE ICC*, 2019, pp. 1–6.

[23] B. C. Chatterjee, N. Sarma, and E. Oki, "Routing and spectrum allocation in elastic optical networks: A tutorial," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 3, pp. 1776–1800, 2015.

[24] J. Comellas and G. Junyent, "Allocating multi-rate requests to single-rate lightpaths in flexgrid optical networks," in *proc. IEEE ICC*, 2018, pp. 1–6.

[25] D. Zheng, E. Guler, C. Peng, G. Luo, L. Tian, and X. Cao, "Dependence-aware service function chain embedding in optical networks," in *proc. IEEE ICC*, 2019, pp. 1–6.

[26] X. Chen, Z. Zhu, R. Proietti, and S. J. B. Yoo, "On incentive-driven vnf service chaining in inter-datacenter elastic optical networks: A hierarchical game-theoretic mechanism," *IEEE Trans. Netw. Serv. Manag.*, 2019.

[27] M. Zhu, Q. Sun, S. Zhang, P. Gao, B. Chen, and J. Gu, "Energy-aware virtual optical network embedding in sliceable-transponder-enabled elastic optical networks," *IEEE Access.*, vol. 7, pp. 41 897–41 912, 2019.

[28] S. Lee and M. Shin, "A self-recovery scheme for service function chaining," in *proc. IEEE ICTC*, 2015.

[29] A. Engelmann and A. Jukan, "A reliability study of parallelized VNF chaining," in *proc. IEEE ICC*, 2018, pp. 1–6.

[30] D. Yamada and N. Shinomiya, "A solving method for computing and network resource minimization problem in service function chain against multiple VNF failures," in *proc. IEEE ICC*, 2019, pp. 30–38.

[31] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *proc. IEEE IWQoS*, 2017, pp. 1–10.

[32] J. Li, W. Liang, M. Huang, and X. Jia, "Providing reliability-aware virtualized network function services for mobile edge computing," in *proc. IEEE ICDCS*, 2019.

[33] Q. Hu, Y. Wang, and X. Cao, "Survivable network virtualization for single facility node failure: A network flow perspective," *Opt. Switch. Netw.*, vol. 10, no. 4, pp. 406–415, 2013.

[34] B. Guo, C. Qiao, J. Wang, H. Yu, Y. Zuo, J. Li, Z. Chen, and Y. He, "Survivable virtual network design and embedding to survive a facility node failure," *IEEE J. Light. Tech.*, 2014.

[35] R. W. Irving, "On Approximating the Minimum Independent Dominating Set," *Inf. Process. Lett.*, vol. 37, no. 4, pp. 197–200, 1991.

[36] Y. Shih, H. Lin, A. Pang, C. Chuang, and C. Chou, "An NFV-based service framework for IoT applications in edge computing environments," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1419–1434, 2019.

[37] S. Barmpounakis, G. Tsiatsios, M. Papadakis, E. Mitsianis, N. Koursioumpas, and N. Alonistioti, "Collision avoidance in 5G using MEC and NFV: The vulnerable road user safety use case," *Comput. Netw.*, vol. 172, p. 107150, 2020.

[38] G. Castellano, F. Esposito, and F. Risso, "A distributed orchestration algorithm for edge computing resources with guarantees," in *proc. IEEE INFOCOM*, 2019, pp. 2548–2556.

[39] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks," *Comput. Netw.*, vol. 166, p. 106980, 2020.

[40] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges,

and implementation in next generation mobile networks (vEPC)," *IEEE Netw.*, vol. 28, no. 6, pp. 18–26, 2014.

[41] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, G. Fu, and Z. Zhu, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access.*, vol. 6, pp. 26 567–26 577, 2018.

[42] G. P. Sharma, W. Tavernier, D. Colle, and M. Pickavet, "VNF-AAPC: Accelerator-aware VNF placement and chaining," *Comput. Netw.*, vol. 177, p. 107329, 2020.

[43] D. Zheng, C. Peng, X. Liao, and X. Cao, "Toward optimal hybrid service function chain embedding in multiaccess edge computing," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6035–6045, 2020.

[44] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A Survey on 5G Networks for the Internet of Things: communication Technologies and Challenges," *IEEE Access.*, vol. 6, pp. 3619–3647, 2018.

[45] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 554–568, 2017.

[46] F. He, T. Sato, and E. Oki, "Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures," in *proc. DRCN*, 2019.

[47] Y. Fouquet, D. Nace, M. Pióro, and M. Poss, "An optimization framework for traffic restoration in optical wireless networks with partial link failures," *Opt. Switc. Netw.*,

vol. 23, pp. 108 – 117, 2017.

[48] S. R. Chowdhury, R. Ahmed, M. M. Alam Khan, N. Shahriar, R. Boutaba, J. Mitra, and F. Zeng, "Dedicated protection for survivable virtual network embedding," *IEEE Trans. Netw. Service Manag.*, 2016.

[49] J. Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao, "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246–2259, 2018.

[50] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue, "Guaranteed-availability network function virtualization with network protection and vnf replication," in *proc. IEEE GLOBECOM*, 2017, pp. 1–6.

[51] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Robust optimization model for backup resource allocation in cloud provider," in *proc. IEEE ICC*, 2018, pp. 1–6.

[52] A. J. Gonzalez, G. Nencioni, A. Kamisiński, B. E. Helvik, and P. E. Heegaard, "Dependability of the NFV orchestrator: State of the art and research challenges," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 3307–3329, 2018.

[53] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surv. Tut.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[54] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, 2019.

[55] Y. Yue, B. Cheng, X. Liu, M. Wang, B. Li, and J. Chen, "Resource optimization and delay guarantee virtual network function placement for mapping sfc requests in cloud networks," *IEEE Trans. Netw. Service Manag.*, pp. 1–16, 2021.

[56] Z. Wang, J. Zhang, T. Huang, and Y. Liu, "Service function chain composition, placement, and assignment in data centers," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1638–1650, 2019.

[57] H. A. Alameddine, C. Assi, M. H. Kamal Tushar, and J. Y. Yu, "Low-latency service schedule orchestration in NFV-based networks," in *proc. IEEE NetSoft*, 2019, pp. 378–386.

[58] I. A. Ouedraogo and E. Oki, "Optimization approach to lower capacity requirements in backup networks," in *Proc. IEEE CQR*, 2016, pp. 1–6.

[59] Y. Hirano, F. He, T. Sato, and E. Oki, "Backup network design against multiple link failures to avoid link capacity overestimation," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 1254–1267, 2020.

[60] S. Petale and J. Thangaraj, "Link failure recovery mechanism in software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1285–1292, 2020.

[61] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, U. Shigeo, and E. Oki, "Robust optimization model for primary and backup resource allocation in cloud providers," *IEEE Trans. Cloud Comput.*, pp. 1–1, 2021.

[62] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "Raba: Resource-aware backup allocation for a chain of virtual network functions," in *proc. IEEE INFOCOM*,

2019, pp. 1918–1926.

[63] X. Shang, Y. Huang, Z. Liu, and Y. Yuan, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," in *proc. IEEE INFOCOM*, 2020, pp. 1–9.

[64] N. Shahriar, M. Zulfiqar, S. R. Chowdhury, S. Taeb, M. Tornatore, R. Boutaba, J. Mitra, and M. Hemmati, "Disruption-minimized re-adaptation of virtual links in elastic optical networks," in *proc. IEEE OFC*, 2020, pp. 1–3.

[65] W. D. Grover, *Mesh-Based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Prentice Hall PTR, 2003.

[66] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[67] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6g networks: Use cases and technologies," *IEEE Commun. Magaz.*, vol. 58, no. 3, pp. 55–61, 2020.

[68] H. Cao, J. Du, H. Zhao, D. X. Luo, N. Kumar, L. Yang, and F. R. Yu, "Resource-ability assisted service function chain embedding and scheduling for 6g networks with virtualization," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3846–3859, 2021.

[69] C. L. Stergiou, K. E. Psannis, and B. B. Gupta, "IoT-based big data secure management in the fog over a 6G wireless network," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5164–5171, 2021.

[70] K. B. Costa, F. S. D. Silva, A. V. Neto, and F. L. Verdi, "Towards edge-cloud-supported monitoring at cloud-network slice granularity," in *proc. IEEE NFV-SDN*, 2021, pp.

161–166.

[71] M. M. Erbati and G. Schiele, "Application- and reliability-aware service function chaining to support low-latency applications in an nfv-enabled network," in *proc. IEEE NFV-SDN*, 2021, pp. 120–123.

[72] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, 2015.

[73] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Networks*, vol. 133, pp. 212–262, 2018.

[74] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," in *proc. IEEE INFOCOM*, 2020, pp. 2096–2105.

[75] W. Wei, H. Gu, W. Deng, Z. Xiao, and X. Ren, "Abl-tc: A lightweight design for network traffic classification empowered by deep learning," *Neurocomputing*, vol. 489, pp. 333–344, 2022.

[76] W. Wei, R. Yang, H. Gu, W. Zhao, C. Chen, and S. Wan, "Multi-objective optimization for resource allocation in vehicular cloud computing networks," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–10, 2021.

[77] F. He, T. Sato, and E. Oki, "Survivable virtual network embedding model with shared protection over elastic optical network," in *proc. IEEE CloudNet*, 2019, pp. 1–3.

[78] C. Peng, D. Zheng, S. Philip, and X. Cao, "Latency-bounded off-site virtual node protection in NFV," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 2545–2556,

2021.

[79] R. Fujita, F. He, and E. Oki, "Shared backup resource assignment for middleboxes considering server protection capabilities," *Comput. Networks*, vol. 186, p. 107734, 2021.

[80] C. Peng, D. Zheng, and X. Cao, "Minimum cost hybrid node protection in NFV," in *proc. IEEE ICCC*, 2021, pp. 345–349.

[81] F. He and E. Oki, "Backup allocation model with probabilistic protection for virtual networks against multiple facility node failures," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 2943–2959, 2021.

[82] C. Peng, D. Zheng, and X. Cao, "Latency-aware VNF protection for network function virtualization in elastic optical networks," in *proc. IEEE GLOBECOM*, 2021, pp. 1–6.

[83] A. Farrel, S. Bryant, and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining," RFC 8595, 2019.

[84] L. Golab, F. Korn, F. Li, B. Saha, and D. Srivastava, "Size-constrained weighted set cover," in *proc. IEEE ICDE*, 2015, pp. 879–890.

[85] K. Jiang, H. Zhou, X. Chen, and H. Zhang, "Mobile edge computing for ultra-reliable and low-latency communications," *IEEE Commun. Stand. Mag.*, vol. 5, no. 2, pp. 68–75, 2021.

[86] B. Hassan, S. Baig, and M. Asif, "Key technologies for ultra-reliable and low-latency communication in 6g," *IEEE Commun. Stand. Mag.*, vol. 5, no. 2, pp. 106–113, 2021.

[87] H. Huang, W. Miao, G. Min, J. Tian, and A. Alamri, "Nfv and blockchain enabled

5g for ultra-reliable and low-latency communications in industry: Architecture and performance evaluation," *IEEE Trans. Ind. Informatics*, vol. 17, no. 8, pp. 5595–5604, 2020.

[88] G. Sun, R. Zhou, J. Sun, H. Yu, and A. V. Vasilakos, "Energy-efficient provisioning for service function chains to support delay-sensitive applications in network function virtualization," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6116–6131, 2020.

[89] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, pp. 138–155, 2016.

[90] H. Chen, R. Abbas, P. Cheng, M. Shirvanimoghaddam, W. Hardjawana, W. Bao, Y. Li, and B. Vucetic, "Ultra-reliable low latency cellular networks: Use cases, challenges and approaches," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 119–125, 2018.

[91] Z. Ye, P. N. Ji, and T. Wang, "Survivable distributed fiber optic sensors placement against single link failure," in *proc. IEEE ICC*, 2021, pp. 1–6.

[92] J. Yim, J. Bang, Y. Nam, Y. Shin, and E. Lee, "Efficient multipath routing protocol against path failures in wireless sensor networks," in *proc. IEEE WMNC*, 2019, pp. 136–140.

[93] F. He and E. Oki, "Backup allocation model with probabilistic protection for virtual networks against multiple facility node failures," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 2943–2959, 2021.

[94] Z. Shu and T. Taleb, "A novel qos framework for network slicing in 5g and beyond networks based on sdn and nfv," *IEEE Netw.*, vol. 34, no. 3, pp. 256–263, 2020.

[95] G. Moualla, T. Turletti, and D. Saucez, "An availability-aware sfc placement algorithm for fat-tree data centers," in *proc. IEEE CloudNet*, 2018, pp. 1–4.

[96] P. K. Thiruvasagam, A. Chakraborty, A. Mathew, and C. S. R. Murthy, "Reliable placement of service function chains and virtual monitoring functions with minimal cost in softwarized 5g networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1491–1507, 2021.

[97] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *proc. IEEE IWQoS*, 2017, pp. 1–10.

[98] Y. Wu, M. Tornatore, S. Ferdousi, and B. Mukherjee, "Green data center placement in optical cloud networks," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 3, pp. 347–357, 2017.

[99] N. Mahala and J. Thangaraj, "Spectrum assignment technique with first-random fit in elastic optical networks," in *proc. IEEE RAIT*, 2018, pp. 1–4.

[100] D. Zheng, G. Shen, X. Cao, and B. Mukherjee, "Towards optimal parallelism-aware service chaining and embedding," *IEEE Tran. Netw. Serv. Manag.*, pp. 1–15, 2022.

[101] M. Ghaznavi, E. Jalalpour, B. Wong, R. Boutaba, and A. J. Mashtizadeh, "Fault tolerant service function chaining," in *proc. ACM SIGCOMM*, 2020, p. 198–210.

[102] J. Xie, R. Zhou, G. Sun, J. Sun, and H. Yu, "Fast recovery for online service function chaining interruption using adaptive migration," *Clust. Comput.*, vol. 25, no. 2, pp. 1321–1339, 2022.

[103] T. Duan and V. Dinavahi, "Fast path recovery for single link failure in sdn-enabled wide

area measurement system," *IEEE Trans. Smart Grid*, vol. 13, no. 2, pp. 1645–1653, 2022.

[104] N. Siasi, M. A. Jasim, A. Yayimli, and N. Ghani, "Service function chain survivability provisioning in fog networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1117–1128, 2022.

[105] D. Xu, Y. Xiong, and C. Qiao, "Novel algorithms for shared segment protection," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 8, pp. 1320–1331, 2003.

[106] D. Zheng, E. Guler, C. Peng, G. Luo, L. Tian, and X. Cao, "Dependence-aware service function chain embedding in optical networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[107] S. Cai, F. Zhou, Z. Zhang, and A. Meddahi, "Disaster-resilient service function chain embedding based on multi-path routing," in *proc. IEEE INFOCOM*, 2021, pp. 1–7.

[108] G. Kibalya, J. Serrat, J.-L. Gorricho, J. Serugunda, and P. Zhang, "A multi-stage graph based algorithm for survivable service function chain orchestration with backup resource sharing," *Comput. Commun.*, vol. 174, pp. 42–60, 2021.

[109] U. Brandes, "A faster algorithm for betweenness centrality," *J. mathe. soc.*, vol. 25, no. 2, pp. 163–177, 2001.