Energy- and Reliability-Aware Provisioning of Parallelized Service Function Chains with Delay Guarantees

Venkatarami Reddy Chintapalli$, *Graduate Student Member, IEEE,* Balaprakasa Rao Killi#, *Senior Member, IEEE,* Rajat Partani⋆, Bheemarjuna Reddy Tamma•, *Senior Member, IEEE*, and C. Siva Ram Murthy•, *Fellow, IEEE*

***Abstract*—Network Functions Virtualization (NFV) leverages virtualization and cloud computing technologies to make networks more flexible, manageable, and scalable. Instead of using traditional hardware middleboxes, NFV uses more flexible Virtual Network Functions (VNFs) running on commodity servers. One of the key challenges in NFV is to ensure strict reliability and low latency while also improving energy efficiency. Any software or hardware failures in an NFV environment can disrupt the service provided by a chain of VNFs, known as a Service Function Chain (SFC), resulting in significant data loss, delays, and wasted resources. Due to the sequential nature of SFC, latency increases linearly with the number of VNFs. To address this issue, researchers have proposed parallelized SFC or VNF parallelization, which allows multiple independent VNFs in an SFC to run in parallel. In this work, we propose a method to solve the parallelized SFC deployment problem as an Integer Linear Program (ILP) that minimizes energy consumption while ensuring reliability and delay constraints. Since the problem is *NP-hard*, we also propose a heuristic scheme named *ERASE* that determines the placement of VNFs and routes traffic through them in a way that minimizes energy consumption while meeting capacity, reliability, and delay requirements. The effectiveness of *ERASE* is evaluated through extensive simulations and it is shown to perform better than benchmark schemes in terms of total energy consumption and reliability achieved.**

***Index Terms*—Network Functions Virtualization, Parallelized Service Function Chaining, Flexible Resource Allocation, Energy Consumption, Reliability.**

I. Introduction

Network Functions Virtualization (NFV) has emerged as a promising solution to overcome the limitations of traditional middleboxes in communication networks. By decoupling hardware and software, NFV enables the deployment of Virtual Network Functions (VNFs) on commodity servers, offering enhanced flexibility, programmability, and cost efficiency [1]. These VNFs are interconnected in a specific order to form Service Function Chains (SFCs), which are vital components of network services provisioned by Cloud Service Providers (CSPs) on NFV infrastructure (NFVI) [1]. While NFV presents numerous benefits, it also introduces new challenges. One such challenge is the considerable energy consumption associated with provisioning VNFs on NFVI. This energy consumption comprises of the energy consumed by the servers hosting various VNF instances and the energy consumed by network devices that route traffic. Existing studies have shown that NFV can reduce energy consumption compared to traditional appliance-based networks [2]. However, further improvements in energy efficiency are necessary to optimize both servers and network devices within NFV-enabled networks. The placement of SFCs in an optimized way to enhance energy efficiency has been identified as one of the key issues in [3]–[5].

Another critical consideration in NFV-based networks is the provision of Ultra-Reliable and Low Latency Communication (URLLC) services. URLLC services are vital for latency-sensitive applications such as autonomous driving, Augmented Reality (AR), Virtual Reality (VR), and industrial control. However, the sequential execution of VNFs in SFCs results in increased propagation and processing delays. Deploying VNFs at the network edge can effectively reduce end-to-end delay [6]. Nonetheless, meeting the stringent latency requirements of URLLC remains a significant challenge in NFV-based networks. To mitigate the impact caused by the processing delay, recent works in [7], [8] demonstrated that VNFs of an SFC could be executed in parallel if the operations of VNFs do not conflict with each other as shown in Fig. 1. For instance, Deep Packet Inspection (DPI) can be executed in parallel with Flow Monitor (FM) as these only inspect packet data without altering its content, whereas DPI and encryption cannot execute in parallel as one of them might modify the packet. In addition, these works shown significant improvement in achieved throughputs along with reduction in latency when the VNFs of an SFC are deployed in parallel in real-world systems. However, the migration from specialized hardware to virtualized software may cause reliability issues. The reliability[1] of an SFC is crucial, as the entire chain must work correctly for the service to function properly. The reliability of an SFC decreases as the chain length increases, even if the reliability of individual VNFs remains high. For

$Venkatarami Reddy Chintapalli is with National Institute of Technology Calicut, India.

#Balaprakasa Rao Killi is with National Institute of Technology Warangal, India.

⋆Rajat Partani is with National Institute of Technology Surathkal, India.

•Bheemarjuna Reddy Tamma and C. Siva Ram Murthy are with Indian Institute of Technology Hyderabad, India.

This research work was supported by the Science and Engineering Research Board, New Delhi, India. Grant number: JBR/2021/000005.

[1]In the context of the core network, the term availability is more commonly used to describe the measure of system's uptime and service accessibility. Therefore, we use the terms availability and reliability interchangeably throughout this work.

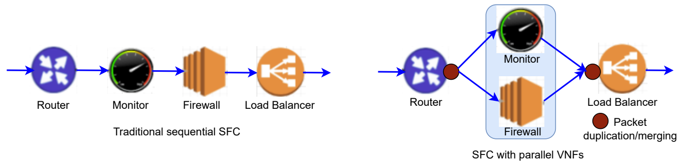


Figure 1: Traditional sequential SFC vs. parallelized SFC [7].

instance, the reliability of an SFC, which contains six different VNFs, is only $0.95^6 = 0.74$, even though the reliability of each VNF is 0.95. Moreover, individual VNFs may fail due to software bugs or hardware faults, further impacting the reliability of the SFC. Therefore, it is essential to devise efficient mechanisms for deploying VNFs in order to improve the reliability of SFCs in NFV-based networks.

Existing works [9], [10] on sequential SFCs are not efficient for meeting the delay and reliability requirements of URLLC services. Therefore, *VNF parallelization* has been proposed as a potential technique [7], [8], but it introduces overheads such as packet duplication/merging and packet deposition problems. To address these overheads, we proposed a novel flexible resource allocation scheme for deploying parallelizable VNFs of an SFC (named such SFC as parallelized SFC (PSFC)) in [11]. However, to the best of our knowledge, none of the existing works considered the energy and reliability aspects while deploying PSFCs in NFV-based networks.

Motivated by this, in this work, we propose a flexible resource allocation-based PSFC deployment scheme named *ERASE* which minimizes the total energy consumption while guaranteeing reliability and end-to-end delay requirements of the network service. *ERASE* focuses on reducing packet duplication and merging overheads by deploying all parallelizable VNFs of a PSFC on a single server. If a server's capacity is insufficient, partial parallelism is explored for VNF deployment. To achieve flexible resource allocation, we utilize a resource-delay dependency model [11], [12], which enables efficient resource assignment to minimize packet deposition and conserve resources for additional requests. Furthermore, we consider both the reliability of physical nodes and VNFs during deployment, creating backup instances when necessary to ensure PSFC reliability. To avoid additional overheads, backup instances for parallelized VNFs are placed on the same server as the primary instances (*i.e., onsite* backup). For *offsite* backup, it prefers deploying backup VNF instances only along the same path as the primary VNFs of PSFCs, thereby reducing bandwidth consumption in the network while easily meeting the end-to-end delay requirements. The primary objective is to minimize the number of servers and network links utilized for deploying PSFCs while satisfying the end-to-end delay and reliability requirements through intelligent utilization of server and network link statuses.

The main contributions of this paper are summarized as follows:

- We present a backup strategy that effectively mitigates VNF parallelization overheads in PSFC deployment.
- We formulate the PSFC deployment problem as an Integer Linear Program (ILP)-based optimization problem to minimize energy consumption while ensuring reliability and meeting delay requirements. The proposed ILP is shown to be NP-hard by reducing it to a well-known bin packing combinatorial optimization problem.
- To overcome the computational complexity of the ILP model, we propose a heuristic solution named *ERASE* which consists of three stages: VNF deployment, VNF routing, and VNF backup deployment. The proposed ILP and the heuristic scheme consider the possibility of deploying PSFC with partial parallelism if a PSFC request with full parallelism cannot be accommodated in the network.
- We extensively evaluate the proposed solution against five benchmark algorithms for parallelized and sequential SFC deployments. The results demonstrate significant energy savings and successful fulfillment of delay and reliability requirements of the PSFCs considered for deployment in NFV-based networks.

The rest of this paper is organized as follows. Section II provides background information on the overheads associated with VNF parallelization and the dependency of VNF processing delay on its resource allocation. The related work is discussed in Section III. In Section IV, we present the system model. The problem of energy- and reliability-aware parallelized SFC placement is formulated in Section V, followed by a polynomial-time heuristic solution to address this problem in Section VI. The performance results are presented in Section VII. Finally, the conclusions are given in Section VIII. Throughout the paper, we use the terms node and server interchangeably.

II. Background

Existing works [7], [8] vividly showcase the application of parallelized SFC in diverse real-world scenarios. For instance, NFP [7] and Parabox [8] exemplify scenarios like real-time analytics and On-line Data-Intensive (OLDI) applications, including web search and online retail, where minimal packet delay is critical. Applications like algorithmic stock trading and high-performance distributed memory caches demand ultra-low latency from cloud infrastructure's VNFs. It was demonstrated that 53.8% of VNF pairs could operate in parallel and it underscores the significant potential of VNF parallelism optimization. VNF parallelization substantially decreases the overall end-to-end delay encountered by VNFs, enabling the fulfillment of stringent delay requirement of services. As emerging services continue to evolve, ensuring reliability alongside delay remains a paramount criterion. Consequently, devising mechanisms to simultaneously fulfill these dual imperatives, while deploying parallelized SFCs with minimized energy consumption, emerges as a crucial requirement for network operators. This work contributes to fulfilling this demand by presenting a PSFC deployment mechanism designed for this purpose.

The parallelization of VNFs in SFCs is based on the operation types of the VNFs. Two VNFs can be parallelized if they both perform only read operations, or if one performs a read operation and the other performs a write operation, as long as they do not operate on the same data area. In many recent works [7], [8], [13]–[16], standard procedures have been proposed for efficiently generating PSFCs for a given set of sequential SFCs. For example, [7] and [8] clearly outlined the steps for parallel conversion of SFCs and provided guidelines on determining VNF parallelism. In this work, we assume that a given set of SFC requests have already undergone the standard procedure for conversion to PSFC requests, as described in the literature.

A PSFC is composed of a series of Parallel Entities (PEs), each consisting of either a set of parallelizable VNFs or a single VNF. Figs. 3a and 3b illustrate two possible PSFC combinations of the sequential SFC presented in Fig. 2. It is important to note that if none of the VNFs in a sequential SFC can run in parallel, all PEs in the resulting PSFC will consist of a single VNF each, maintaining the same structure as the original sequential SFC. Fig. 3b represents the scenario where no VNFs can run in parallel, resulting in a PSFC that is identical to the original sequential SFC (i.e., Fig. 2). The blue dashed rectangle represents a PE. Parallel processing of packets by different VNFs can bring significant latency benefits to many emerging latency-sensitive applications. However, deploying these parallelizable VNFs on the network introduces packet duplication/merging overhead and packet deposition problems.

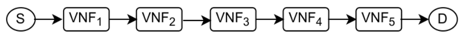
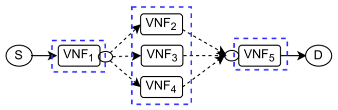
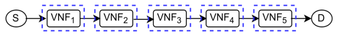


Figure 2: Example of sequential SFC request. S and D represent the source and destination, respectively.



(a) PSFC with one PE with multiple VNFs



(b) PSFC with all PEs with single VNF

Figure 3: Example of different PSFCs.

A. Packet Duplication/Merging Overhead

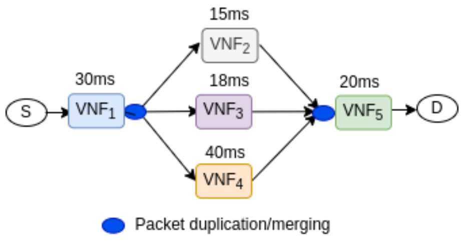
The packet duplication and merging modules are used to copy packets to multiple VNFs running in parallel and then merge the results of the parallel processing back into a single packet stream. This allows for parallelization of the SFC in the network. However, it introduces additional overhead in terms of network resources and computation, which can potentially lead to increased latency and reduced overall network performance. A few studies have investigated methods for minimizing this overhead. One approach proposed in works [7] and [8] is to deploy all PEs of a PSFC on the same server and share the same memory, thereby avoiding the expensive overhead of duplicating and merging the whole packet payload. However, this approach can lead to unused resource fragmentation. Studies such as [13] and [18] have found that when parallel VNFs are deployed on the same server, the intra-server packet duplication/merging overheads are negligible compared to the processing delay of VNFs. However, when parallelizable VNFs are deployed on different servers, inter-server packet duplication/merging overheads are significant. Therefore, in this work, it is proposed to deploy all parallelizable VNFs on the same server to eliminate inter-server packet duplication/merging overheads. If it is determined that none of the servers possess adequate capacity to deploy all parallel VNFs of a PE, alternative solutions are explored, such as converting full VNF parallelism to partial VNF parallelism. This entails dividing such PE into multiple PEs, with the number of VNFs on the resultant PEs being reduced, in order to decrease the required resources and increase the probability of identifying suitable nodes. However, it should be noted that this conversion may result in an increase in the overall latency of the PSFC due to the sequential execution of certain VNFs. The details of conversion from full VNF parallelism to partial VNF parallelism are explained in Section VI-A3.

B. Packet Deposition Problem

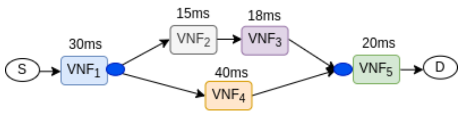
A significant imbalance in the processing delays of parallelizable VNFs causes packet deposition, i.e., packets going through faster VNFs need to be cached in the merging module and wait for the packets going through other slower VNFs. Consider a PSFC request presented in Fig. 4a wherein VNF_2, VNF_3, and VNF_4 can be executed in parallel. All packets leaving VNF_1 must be duplicated and sent to each parallel VNF. Then, the processed packets are merged in the merging node located just before VNF_5. The value at each VNF represents its processing delay. We can observe that the processing delays of VNF_2 and VNF_3 are less than that of VNF_4. Therefore, the packets processed by VNF_2 and VNF_3 reach the merging module sooner than the packet copies processed by VNF_4. The merging module has to allocate additional memory resources to cache the early-arrived packets. A significant imbalance in procesing delays between parallel VNFs results in higher memory resource consumption. Recently, in [11], [17], the authors proposed two approaches to reduce the imbalance in processing delays between parallel VNFs in PSFC requests. The first approach is to merge some of the parallel VNFs and run them sequentially [17]. This reduces the delay difference between the two parallel branches and reduces the number of packets cached and memory resources required for packet caching in the merging module. For example, merging VNF_2 and VNF_3 in Fig. 4a into one branch significantly reduces the delay difference between the two parallel branches, as shown in Fig. 4b. Recently, in [11], we proposed an improved approach that scales down the resources of VNFs in a PE using their resource-delay dependencies so that they process packets with a delay approximately equal to the peak VNF delay of the PE as shown in Fig. 4c. This avoids over-provisioning of resources to VNFs and saves resources which can be used to deploy or accept more PSFC requests.

C. VNF Resource-Delay Dependency

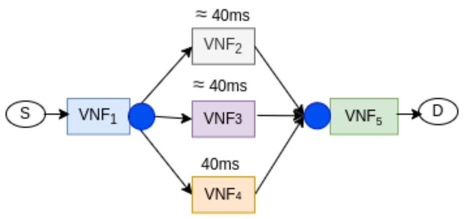
The impact of allocated resources on processing delay is captured using Fig. 5. Let ζ^q_{min} and ζ^q_{max} be the minimum and the maximum processing delay of a VNF instance q, respectively. We can observe from the figure that ζ^q_{min} and ζ^q_{max} are reached by allocating resources η^q_{max} and η^q_{min}, respectively. For any amount of resources higher than η^q_{max}, the delay reaches its lower bound ζ^q_{min} and cannot be further improved. Moreover, for an amount of resources less than η^q_{min}, the VNF q fails to



(a) PSFC request



(b) Using APPM [17]



(c) Using FlexSFC [11]

Figure 4: Illustration of packet deposition problem.

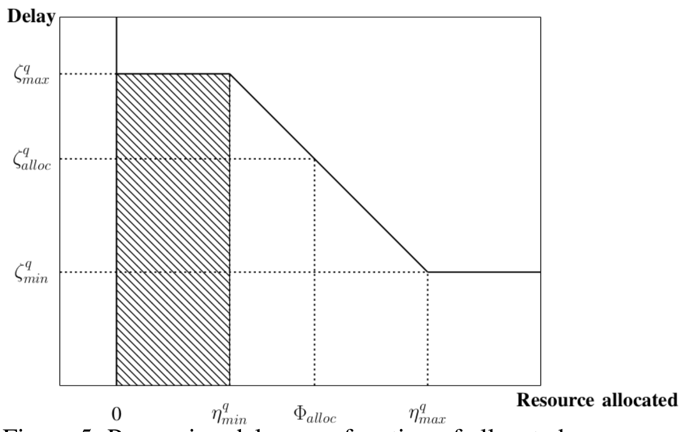


Figure 5: Processing delay as a function of allocated resources (linear dependency case) [12].

execute. Recent studies [11], [12], [19] verified this behavior by conducting experiments on different VNFs. Therefore, the processing delay of any VNF q (τ_q) is a linear function of system resources allocated (Φ_{alloc}) to it within a certain range of resources [12]. This linear function is captured using the following equation:

$$\tau_q = h(\Phi_{alloc}) = a \times \Phi_{alloc} + b \tag{1}$$

where a and b represent slope and intercept which are given by Eq. (2) and Eq. (3), respectively.

$$a = \frac{\zeta^q_{max} - \zeta^q_{min}}{\eta^q_{min} - \eta^q_{max}} \tag{2}$$

$$b = \frac{(\zeta^q_{min} \times \eta^q_{min}) - (\zeta^q_{max} \times \eta^q_{max})}{\eta^q_{min} - \eta^q_{max}} \tag{3}$$

The processing delay of the j^{th} PE of the i^{th} PSFC is represented by $PD_{i,j}$. The processing delay of the PE is determined by the VNF with the highest delay, and the delays caused by other VNFs can be disregarded.

$$PD_{i,j} = \max_{\{q \in P_i\}} \tau_q \tag{4}$$

It is important to note that all VNFs within a PE should be placed on the same node to avoid the additional costs of duplicating and merging packets. Therefore, all VNFs in a PE will adjust their resources (scale down) to match their processing delays to the delay $PD_{i,j}$ of the PE using Eqs. 1, 2, and 3.

III. Related Work

A. Traditional SFC placement

The SFC placement problem has been widely researched, with a focus on various design goals, including reducing delays [12], [20], maximizing network throughput [21], [22], minimizing energy consumption [3]–[5], [9], [10], and increasing reliability [23]–[25]. Many studies have aimed to address these goals individually [3], [12], [20], [23]–[25], while others [9], [10] have sought to address them simultaneously. Studies [23]–[27] have focused on guaranteeing the reliability of SFCs, with the majority of them focusing on using redundancy as a way to improve reliability. For example, in [23], the authors proposed different schemes that aim to minimize the number of backups required while still meeting reliability requirements. Very few works [26], [27] considered both software (i.e., VNF) and hardware (i.e., Physical Node (PN)) reliability while deploying VNFs in the network topology. Nevertheless, most of the research focused on sequential SFCs and paid little attention to VNF parallelization.

To get the latency benefits, in [28]–[30], the authors explored VNF multi-feature deployment where multiple VNFs of same SFC are collocated within the same virtual machine (VM) or server enabling the avoidance of packet copy operations between multiple VNFs. Additionally, this consolidation strategy eliminates inter-VNF communication overhead that may arise if the VNFs are deployed on different servers. This approach can enhance overall efficiency and reduce latency by avoiding inter-node communication. However, it is crucial to note that even though all the VNFs are deployed on the same node in multi-feature deployment, they are still executed sequentially in the existing works. Consequently, they did not fully exploit the benefits of VNF parallelization, particularly in terms of latency reduction.

B. Parallelized SFC placement

Recent studies [7], [8], [32] demonstrated the feasibility and effectiveness of VNF parallelization, which enables the parallel execution of multiple VNFs and can lead to significant reductions in delay compared to traditional SFCs. The observation in [7] indicated that 53.8% of VNF pairs in enterprise networks could work in parallel. These findings have garnered significant attention from the academic community. Due to the great advantage of delay reduction, the parallel VNFs were preferred to the traditional sequential ones when deploying SFCs [13], [14], [16], [17], [32]. Most of these

Table I: Comparison of the characteristics of the existing works that are closest to the proposed work

Work	SFC Placement	VNF Parallelization	Packet Decomposition Problem	VNF Resource-delay Dependency	Energy-aware	VNF Reliability	PN Reliability	End-to-End Delay
[3]	✓	×	×	×	✓	×	×	✓
[5]	✓	×	×	×	✓	×	×	✓
[20]	✓	×	×	✓	×	×	×	✓
[11]	✓	✓	✓	✓	×	×	×	✓
[12]	✓	×	×	✓	×	×	×	✓
[14]	✓	✓	×	×	×	×	×	×
[13]	✓	✓	×	×	×	×	×	×
[17]	✓	✓	✓	×	×	×	×	×
[18]	×	✓	×	×	×	×	×	×
[24]	✓	×	×	×	×	✓	×	✓
[26]	✓	×	×	×	×	✓	✓	×
[31]	✓	✓	×	×	×	×	×	✓
[16]	✓	✓	×	×	×	×	×	✓
[32]	✓	✓	×	×	×	×	×	×
[33]	✓	✓	×	×	×	✓	×	✓
Proposed work	✓	✓	✓	✓	✓	✓	✓	✓

studies have not addressed the overhead issues introduced by parallelizing VNFs, such as packet duplication/merging and packet deposition problems. A limited number of studies on parallelized SFC deployment considered either packet duplication/merging [11], [13], [18] or packet deposition problems [11], [17], but not both.

None of the aforementioned works considered the impact of the amount of allocated resources to a given VNF on its processing delay. A few existing works [12], [20] considered resource-delay dependency while deploying SFCs in the network but these works are on traditional sequential SFC deployments. In our recent work [11], we proposed a scheme, called *FlexSFC*, which uses the resource-delay dependency of VNFs to avoid the parallelization overheads while deploying PSFC requests. However, energy and reliability aspects were not considered while placing PSFCs in the network. In [33], the authors proposed a mechanism to increase the end-to-end service reliability of a parallelized SFC.

The deployment of parallel VNFs requires careful consideration to manage bandwidth consumption effectively. If parallel VNFs are deployed on the same server, the overhead is relatively low and intra-copy/merge costs are quite negligible, as shown in [7] and [13]. This approach ensures efficient resource utilization and reduces additional bandwidth consumption. In contrast, deploying parallel VNFs on different nodes can lead to transmitting multiple copies of the same packet separately, resulting in increased bandwidth consumption compared to sequential SFCs or deploying parallel VNFs on the same node. Moreover, inter-node packet copy/merge delays may impact the latency benefits gained from VNF parallelization [13], [18]. Therefore, the network operators should carefully consider these trade-offs between reduction in SFC latency and increase in bandwidth consumption when deploying parallelizable VNFs in the network.

Differing from the existing works, in this work we consider the deployment of parallelized SFCs with the objective of minimizing total energy consumption while guaranteeing the reliability and delay requirements in NFV-based networks. We also consider the relationship between allocated resources and VNF performance, to avoid parallelization overheads. A comparison of different state-of-the-art solutions with respect to this work is summarized in Table I. To the best of the authors' knowledge, this paper is the first work to investigate the parallelized SFC deployment problem with the objective of minimizing total energy consumption while ensuring reliability and delay requirements in an NFV-based network environment.

IV. System Model

This section describes the models used to represent the physical network and PSFC in our study, followed by the energy and reliability models, and a discussion of different backup strategies that can be used to ensure the reliability of the system.

A. Network Model

The physical network, over which different VNFs of PSFCs need to be deployed, can be modeled as an undirected graph $G = (N, E)$ where N represents the set of Physical Nodes (PNs) in the network and E is the set of links interconnecting the PNs. PN k ($k \in N$) is a server connected through a switch in the network, and its computing capacity and reliability are represented by C_k and R_k, respectively. The reliability of a PN is characterized in terms of Mean Time Between Failures (MTBF). The delay and bandwidth of a link $e \in E$ are represented by D_e and BW_e, respectively.

B. PSFC Model

Let S be the set of PSFC requests. PSFC is an ordered set of PEs connected to each other logically, and it is represented by a tuple $(s_i, d_i, D_i^{req}, BW_i^{req}, P_i, R_i^{req})$. $s_i, d_i \in N$ are the source and destination nodes of i^{th} PSFC request. D_i^{req} and BW_i^{req} are delay and bandwidth requirements of PSFC i, respectively. $P_i = (P_i^1 \rightarrow P_i^2 \rightarrow \cdots \rightarrow P_i^j)$ represents the ordered set of PEs of PSFC i, where each PE has one or more number of VNFs in it which can process incoming packets in parallel. Let c_q, τ_q, and R_q be the computing capacity, processing delay, and reliability of VNF q, respectively. Each PSFC i has its own on-demand reliability requirement R_i^{req} (where $0 < R_i^{req} < 1$).

C. Energy Model

In NFV-enabled networks, PNs and physical links carrying traffic consume electrical energy. In this study, we present a model that looks at the energy used by the servers that host VNFs and the physical links that carry traffic. The energy consumption of a server k is divided into two parts [34]: the energy needed to keep the server running and the energy needed to process network service requests. The first part, E_b^k, is the energy needed to keep the server active. The second part, $load_k$ and E_h^k, is related to how much computing is used and the energy needed to process the requests. Then the energy consumption (E_k) of a server k can be calculated as follows:

$$E_k = E_b^k + (E_h^k - E_b^k) \times (\frac{load_k}{C_k}) \quad \forall k \epsilon N \tag{5}$$

The energy consumption of a physical link depends on its on/off state and bandwidth utilization [35]. Let E_b^e denote the startup energy consumption of link e and E_h^e denote its peak-load energy consumption. The bandwidth consumption of link is denoted as $load_e$. Then the energy consumption of a link (E_e) can be calculated as follows:

$$E_e = E_b^e + (E_h^e - E_b^e) \times (\frac{load_e}{B_e}) \quad \forall e \epsilon E \tag{6}$$

The total energy consumption is expressed as the sum of energy consumption of all running services as follows:

$$E_{total} = \sum_{k \in N} E_k + \sum_{e \in E} E_e \tag{7}$$

It is important to note that, in this paper, we leverage VNF resource-delay dependency [11], [12] to allocate resources flexibly for parallelizable VNFs in order to overcome the packet deposition problem, bringing in significant resource savings, thus, reducing energy consumption proportionally.

D. Reliability Model for PSFC

According to ETSI [36], the reliability of a VNF is defined as the probability that the VNF successfully completes the processing of its intended service. The execution of a VNF may be interrupted due to hardware or software failures (e.g., unexpected restart/shutdown of a PN, network disconnections, software bugs, etc) [24]. Therefore, the reliability of a VNF running on a PN must be expressed by both software and hardware reliability. In this paper, we consider both software and hardware reliabilities, i.e., the reliability of VNFs and PNs. We assume that the switches connecting PNs are absolutely reliable. The reliability of a VNF instance q hosted on PN k is given by

$$R_q^k = R_q \, R_k \tag{8}$$

where R_q represents the reliability of VNF q and R_k represents the hardware reliability of the PN k. A PSFC consists of a series of PEs, each composed of either a set of parallelizable VNFs or a single VNF. It is important to remember that we deploy all VNFs of a PE on the same PN. The reliability of a PSFC is calculated as the product of the reliability of PEs comprising it. The reliability of a PE is presented in Eq. 9.

$$R_{PE} = R_k \prod_{q \in PE} R_q \tag{9}$$

Operators need to deploy primary VNFs reasonably to improve PSFC reliability as much as possible. This can be achieved by picking highly reliable PNs for deploying VNFs. However, merely mapping primary VNFs onto highly reliable PNs is not enough to achieve high reliability. Therefore, PSFC reliability should be improved further by adding redundant backups. Given a set of PSFC requests, each with a specific reliability requirement, we need to determine the minimum number of backup VNFs needed for each PSFC to achieve reliability requirements. We assume at most one backup instance for any VNF can be deployed. It is important to decide the placement of the backup instance for a given VNF. Generally, there are two mechanisms for achieving this, namely, *onsite* backup strategy and *offsite* backup strategy. In *onsite* backup strategy, a backup instance for a VNF is deployed on the same PN where the primary VNF is running. This strategy reduces the overhead of synchronizing the states of primary VNF and backup VNF and saves bandwidth resources. In *offsite* backup strategy, a backup instance is deployed on a different PN. When choosing a PN for an *offsite* backup strategy, it is important to consider the end-to-end delay and the need to reserve bandwidth resources along the path containing the backup instance.

For PEs with multiple VNFs, creating a backup instance for any of their VNFs on another server incurs packet duplication/merging overheads and packet deposition problems. This study suggests a new backup strategy for PSFC to avoid this. It works as follows: For PEs with a single VNF, we proritize deploying the backup instance on the same node as the primary instance (*onsite* backup strategy), given that the node has sufficient capacity. In the cases where the capacity of the primary node is insufficient, we deploy the backup instance on the other node (*offsite* backup strategy). It is crucial to note that PEs consisting of a single VNF do not incur any VNF parallelization overheads. On the other hand, for PEs with multiple VNFs, deploying backup instances on a different node introduces significant inter-node packet copy and merge delays, as highlighted in [18], [32]. This work always strives to deploy backup instances on the same node where the primary VNFs of that PE are deployed. The reason for this is that deploying primary and backup VNF instances on the same node incurs only intra-node packet duplication and merging delays, which are quite negligible as reported in [7], [13]. Moreover, to mitigate the packet deposition problem, we scale down the resources of certain parallel VNFs to minimize the delay discrepancy among them.

Reliability of a PE (with size one) after deploying a backup instance using *onsite* and *offsite* backup strategies are given below, respectively:

$$R_{PE}^B = R_k(1 - (1 - R_q)(1 - R_q)) \tag{10}$$

$$R_{PE}^B = 1 - (1 - R_q R_k)(1 - R_q R_{bn}) \tag{11}$$

where R_{bn} is the reliability of a PN where backup for VNF is deployed.

For each PE with size greater than one, we create backups for a subset of VNFs that results in maximum reliability for PE. Note that, when it is required to deploy a backup for a subset of VNFs in PE to improve the overall PSFC reliability, we choose the less reliable VNFs and deploy backup instances for them on the physical node [25].

$$R^B_{PE} = R_k \max_{S \subseteq PE} \prod_{q \in PE \setminus S} R_q \prod_{t \in S} (1 - (1 - R_t)(1 - R_t)) \quad (12)$$

The reliability of a PSFC after creating backups for PEs is given below:

$$R^B_{PSFC} = \prod_{PE \in PSFC} R^B_{PE} \quad (13)$$

In this work, we made an assumption that the backup VNFs operate in active-standby (specifically, warm-standby) mode [25], [37], where we reserve the same amount of resources as the corresponding primary VNF but with a basic resource consumption (BRC) always in use to each backup VNF instance. The resource consumption of a VNF can generally be divided into two parts. The first part is the virtualization overheads [38], which include the resource consumption required to maintain the image and related libraries of a VNF. This component is referred to as the BRC [39]. The second part is the resource consumption during the VNF's operational state, known as the Duty Resource Consumption (DRC) [37]. By employing active-standby backup, in case of onsite backup strategy (i.e., primary and backup instances are on the same node), it is ensured that backup VNF instance remains synchronized with the corresponding primary VNF. It is worth mentioning that the major portion of the resources allocated for backup instances are utilized only in the event of a primary instance's failure. Therefore, when calculating energy consumption, we did not consider the resources reserved for backup instances.

When dealing with stateful VNFs, it is important to allocate memory for storing the flow states. This memory is needed to copy the states to the backup VNF in case the primary VNF fails. There are different approaches to managing this memory. For example, it can be maintained at the same node where the backup VNF is located or stored in a common repository for all backup VNF instances. As in [26], [27], [33], we also did not consider this aspect in this work. Analyzing the trade-offs associated with memory allocation and exploring its impact on PSFC deployment is an interesting research problem.

V. Problem Formulation

In an NFV-enabled network, the service provider needs to optimally deploy the PSFC requests while meeting various requirements. Specifically, given a physical network G, a set of SFC requests, and their corresponding PSFC combinations (complete/partial parallelism among parallelized VNFs), the service provider needs to determine the following: PSFC combination to be deployed for each SFC, the placement and routing of VNFs in corresponding PSFC combination, and allocating computing resources of underlying physical nodes to corresponding VNFs. Existing works in the literature proposed standard procedures for efficiently generating PSFCs for a given sequential SFC [7], [8], [14], [16].

We define binary decision variables $x^{j,k}_{i,r}$, $y^{q,k}_{i,r,j}$, and $w^e_{i,r,j}$ $\forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall q \in P_{irj}, \forall k \in N, \forall e \in E$. The decision variable $x^{j,k}_{i,r}$ is set to one if the j^{th} PE of r^{th} parallel combination from i^{th} SFC is deployed on node $k \in N$, otherwise it is set to zero. The decision variable $w^e_{i,r,j}$ is set to one if the virtual link between j^{th} and $j+1^{th}$ PEs of r^{th} parallel combination from i^{th} SFC is mapped to physical link $e \in E$, otherwise it is set to zero. The decision variable $y^{q,k}_{i,r,j}$ is set to one if the backup for q^{th} VNF of j^{th} PE of r^{th} parallel combination from i^{th} SFC is deployed on node $k \in N$, otherwise it is set to zero. We also define two binary decision variables $z_k, \forall k \in N$ and $h_e, \forall e \in E$. The decision variable z_k is set to one if the physical node $k \in N$ is active, otherwise it is set to zero. The decision variable h_e is set to one if the physical link $e \in E$ is active, otherwise it is set to zero. Further, we define a decision variables $t_{i,r}$, $\forall i \in S, \forall r \in P_i$. It is set to one if all PEs of r^{th} parallel combination from i^{th} SFC are deployed on some physical node, it is set to zero if none of the PEs of r^{th} parallel combination from i^{th} SFC is deployed on some physical node. All the notations used in this section are given in Table. II.

$$x^{j,k}_{i,r} = \begin{cases} 1 & \text{If } j^{th} \text{ PE of } r^{th} \text{ parallel combination from } i^{th} \\ & \text{SFC is deployed on node } k \\ 0 & \text{Otherwise} \end{cases} \quad (14)$$

$$t_{i,r} = \begin{cases} 1 & \text{If all PEs of } r^{th} \text{ parallel combination from } i^{th} \\ & \text{SFC are deployed on some physical node } k, \\ & \text{i.e., } \sum_{k \in N} \sum_{j \in Pir} x^{j,k}_{i,r} = |P_{ir}| \\ 0 & \text{If } \sum_{k \in N} \sum_{j \in Pir} x^{j,k}_{i,r} = 0 \end{cases} \quad (15)$$

$$y^{q,k}_{i,r,j} = \begin{cases} 1 & \text{If the backup for } q^{th} \text{ VNF of } j^{th} \text{ PE of } r^{th} \text{ parallel} \\ & \text{combination from } i^{th} \text{ SFC is deployed on node } k \\ 0 & \text{Otherwise} \end{cases} \quad (16)$$

$$g_{i,r,j} = \begin{cases} 1 & \text{If backup is created for at least one VNF from } j^{th} \\ & \text{PE of } r^{th} \text{ parallel combination from } i^{th} \text{ SFC} \\ 0 & \text{Otherwise} \end{cases} \quad (17)$$

$$u^{j,k}_{i,r} = \begin{cases} 1 & \text{If backup is created for at least one VNF from } j^{th} \\ & \text{PE of } r^{th} \text{ parallel combination from } i^{th} \text{ SFC and} \\ & \text{those backup instances are deployed on node } k \\ 0 & \text{Otherwise} \end{cases} \quad (18)$$

$$z_k = \begin{cases} 1 & \text{If the physical node } k \in N \text{ is active} \\ 0 & \text{Otherwise} \end{cases} \quad (19)$$

$$w^e_{i,r,j} = \begin{cases} 1 & \text{If the virtual link between } j^{th} \text{ and } j+1^{th} \text{ PEs of} \\ & r^{th} \text{ parallel combination from } i^{th} \text{ SFC is mapped} \\ & \text{to physical link } e \\ 0 & \text{Otherwise} \end{cases} \quad (20)$$

Table II: Glossary for the optimization model

Notation	Description	Notation	Description
N	Set of physical nodes	E	Set of physical links
S	Set of PSFC requests	P_{ir}	Set of PEs in r^{th} parallel combination of i^{th} PSFC
s_i	Source node of i^{th} PSFC	d_i	Destination node of i^{th} PSFC
C_k	Processing capacity of k^{th} physical node	BW_e	Bandwidth of physical link e
R_i^{req}	Reliability requirement of i^{th} PSFC	R_s^q	Reliability of VNF q
R_h^k	Reliability of physical node $k \in N$	D_i^{req}	Delay requirement of i^{th} PSFC
$BW_{i,j}^{req}$	Bandwidth requirement of virtual link between j^{th} and $j+1^{th}$ PEs in i^{th} SFC	$C_{i,j}^{req}$	Processing capacity requirement of j^{th} PE of i^{th} PSFC

$$h_e = \begin{cases} 1 & \text{If the physical link } e \in E \text{ is active} \\ 0 & \text{Otherwise} \end{cases} \tag{21}$$

Our goal is to compute the optimal deployment of PSFC requests so as to minimize energy consumption while ensuring the reliability and delay requirement of each PSFC request. It is formulated as follows:

$$\min E_{total} = \sum_{k\in N} E_k z_k + \sum_{e\in E} E_e h_e + \sum_{i\in S}\sum_{r\in P_i}\sum_{\substack{j\in P_{ir},\\ |P_{irj}|=1}}\sum_{q\in P_{irj}}\sum_{k\in N}\left(|x_{i,r}^{j,k} - y_{i,r,j}^{q,k}|(\sum_{k'\in N} y_{i,r,j}^{q,k'})\right) \tag{22}$$

subject to:

$$\sum_{k\in N} x_{i,r}^{j,k} \leq 1, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir} \tag{23}$$

$$\sum_{k\in N} y_{i,r,j}^{q,k} \leq 1, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall q \in P_{irj} \tag{24}$$

$$\sum_{k\in N} u_{i,r}^{j,k} \leq 1, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir} \tag{25}$$

$$1 - \sum_{q\in P_{ir}^j} y_{i,r,j}^{q,k} \leq Q\, v_{i,r}^{j,k}, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{26}$$

$$\sum_{q\in P_{irj}} y_{i,r,j}^{q,k} - 1 \leq Q\, v_{i,r}^{j,k}, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{27}$$

$$u_{i,r}^{j,k} \leq 1, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{28}$$

$$u_{i,r}^{j,k} \leq \sum_{q\in P_{irj}} y_{i,r,j}^{q,k}, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{29}$$

$$u_{i,r}^{j,k} \geq 1 - Q(1 - v_{i,r}^{j,k}), \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{30}$$

$$u_{i,r}^{j,k} \geq \sum_{q\in P_{irj}} y_{i,r,j}^{q,k} - Q(1 - v_{i,r}^{j,k}), \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{31}$$

$$x_{i,r}^{j,k} \leq z_k, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall k \in N \tag{32}$$

$$u_{i,r}^{j,k} \leq z_k, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall q \in P_{irj}, \forall k \in N \tag{33}$$

$$u_{i,r}^{j,k} = g_{i,r,j} x_{i,r}^{j,k}, \\ \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, |P_{irj}| > 1, \forall q \in P_{irj}, \forall k \in N \tag{34}$$

$$\sum_{r\in P_i} t_{i,r} = 1, \forall i \in S \tag{35}$$

$$t_{i,r} \geq t_{i,r'}, \forall i \in S, r \in P_i, r' \in P_i, |P_{ir}| \leq |P_{ir'}| \tag{36}$$

$$\sum_{i\in S}\sum_{r\in P_i}\sum_{j\in P_{ir}} x_{i,r}^{j,k} C_{i,j}^{req} + \sum_{i\in S}\sum_{r\in P_i}\sum_{j\in P_{ir}}\sum_{q\in P_{irj}} y_{i,r,j}^{q,k} C_{i,j}^{req} \leq C_k, \forall k \in N \tag{37}$$

$$w_{i,r,j}^{e} \leq h_e, \forall i \in S, \forall r \in P_i, \forall j \in P_{ir}, \forall e \in E \tag{38}$$

$$\sum_{i\in S}\sum_{r\in P_i}\sum_{j\in P_{ir}} w_{i,r}^{j,e} BW_{i,j}^{req} \leq B_e, \forall e \in E \tag{39}$$

$$R_i^B \geq R_i^{req}, \forall i \in S \tag{40}$$

$$\sum_{j\in P_i} PD_{i,j} + \sum_{l\in p_{sc}^i} T(l) \leq D_i^{req}, \forall i \in S \tag{41}$$

$$x_{i,r}^{j,k}, g_{i,r,j}, y_{i,r,j}^{q,k}, u_{i,r}^{j,k}, w_{i,r,j}^{e}, t_{i,r}, z_k, h_e \in \{0,1\} \tag{42}$$

First two terms in the objective function describe node and link energy consumption, respectively. Eq. 23 ensures that the primary deployment of each PE of an PSFC is on at most one physical node. Eq. 24 guarantees that the backup deployment of each VNF in a PE of an PSFC is deployed on at most one physical node. Eq. 25 ensures that the backup instances created for VNFs in a PE of an PSFC should be deployed on at most one physical node. Let Q be a constant such that $1, \sum_{q\in P_{irj}} y_{i,r,j}^{q,k} \leq Q$ in any reasonable solution to the problem. Eq. 26-Eq. 27 enforce that $u_{i,r}^{j,k}$ is equal to one if $1 < \sum_{q\in P_{irj}} y_{i,r,j}^{q,k}$, is equal to zero if $\sum_{q\in P_{irj}} y_{i,r,j}^{q,k} > 1$, and could equal either if $1 = \sum_{q\in P_{irj}} y_{i,r,j}^{q,k}$. Eq. 26-Eq. 31 all together enforce the definition of $u_{i,r,j}^{k}$ described in Eq. 18. That is, Eq. 26-Eq. 31 all together ensure that $u_{i,r,j}^{k} = min(1, \sum_{q\in P_{irj}} y_{i,r,j}^{q,k})$. Eq. 32 and Eq. 33 guarantee that primary and/or backup instances of a PE should be deployed on an active server. If the server is not active, then activate it before hosting primary and/or backup instances of a PE. Eq. 34 ensures if a backup is created for a VNF in a PE with size greater than one, then the backup should be deployed on the physical node where the primary instance for PE is deployed. The third term in the objective function ensures that preference is given to the onsite strategy over the offsite strategy when a backup is created for a PE with size 1 due to reliability constraints. The last summand can only take values between 0 and 1 because of the constraint in Eq. 24. If the objectives are met by setting $\sum_{k'\in N} y_{i,r,j}^{q,k'} = 0$, then we do not require a backup for the PE when deployed on node k. However, if we are forced to have $\sum_{k'\in N} y_{i,r,j}^{q,k'} = 1$ (a backup is required), then the multiplicative factor of $x_{i,r}^{j,k} - y_{i,r,j}^{q,k}$ gives preference towards deploying the backup on the same node (onsite strategy). For every SFC $i \in S$, Eq. 35 ensures at most one PSFC is selected and all of its PEs are deployed. For every SFC $i \in S$, Eq. 36 ensures that the preference should be given to PSFC combinations of SFC i with lesser number of PEs. This preference ensures that PSFCs with higher parallelism are favored over those with lower parallelism. It is worth noting that a PSFC with a lesser number of PEs indicates a higher number of VNFs being executed in parallel, in contrast to a PSFC with a higher number of PEs. For each physical node $k \in N$, Eq. 37 guarantees that the sum of processing capacity requirements of VNFs whose primary and backup instances are deployed on physical node k should not exceed its processing capacity. For each physical link, $e \in E$, Eq. 39 guarantees that the sum of bandwidth requirements of virtual links that are mapped to physical link e should not exceed the bandwidth of link e. Eq. 40 ensures that the reliability of PSFC i ($\forall i \in S$), after deploying backup instances for its VNFs, denoted as R_i^B, meets or surpasses the PSFC i reliability requirement, denoted as R_i^{req}. Note that the reliability of a PSFC after deploying backup instances for its VNFs is represented in Eq. 13. Eq. 41 guarantees that the end-to-end delay of the PSFC does not exceed the PSFC delay requirement. The end-to-end delay of a packet traversing a PSFC is defined as the sum of VNF processing delays and propagation delays in the deployed path, where $PD_{i,j}$ represents the processing delay of PE j and P_i represents the set of PEs in PSFC i. p_{sc}^i denotes the path traversed by each deployed PE and $T(l)$ indicates the link delay of link l. Eq. 42 ensures that the decision variables are binary, *i.e.*, take a value of zero or one.

Theorem 1: *PSFC deployment problem is NP-hard.*

Proof: In this, we demonstrate that the SFC placement problem is equivalent to the bin packing problem, which is a well-known *NP-hard* combinatorial optimization problem [40]. The bin packing problem consists of packing a set of items, each with an integer weight, into a set of identical bins, each with an integer capacity c, in such a way that the maximum capacity of any bin is not exceeded while minimizing the number of bins used. To prove that the SFC placement problem is *NP-hard*, it is sufficient to show that an instance of the bin packing problem can be transformed into an instance of the SFC placement problem in polynomial time.

The transformation process is as follows: i) each item in the bin packing problem is considered as an SFC in the SFC placement problem, ii) the integer weight of each item is set equal to the resource requirement of each SFC, iii) the total number of available bins is considered as the total number of substrate nodes, iv) the capacity of each bin is set equal to the resource availability in each substrate node, and v) each item is placed in only one bin, as an SFC is placed in only one substrate node. Since this transformation can be completed in polynomial time with respect to the input size, it can be inferred that the SFC placement problem is *NP-hard*. As the SFC placement problem is a restricted version of the PSFC deployment problem, the PSFC deployment problem is also *NP-hard*. Hence, in the following section, a heuristic approach is proposed to efficiently solve the problem for large instances.

VI. Energy and Reliability Aware Parallelized SFC Placement Mechanism

In this section, we propose an efficient energy-aware parallelized SFC placement algorithm named *ERASE* that guarantees reliability and delay requirements. Algorithm 1 provides an overview of *ERASE*. It takes a network topology G, a set of PSFC requests M, VNFs' resource-delay dependency table as inputs, and returns PSFC placement as the output. VNF resource-delay dependency table contains information of maximum and minimum resources allocated to a VNF and its corresponding delays. Step 4 sorts the PSFCs in non-decreasing order based on the total resources required. Then, for each PSFC, the resources for VNFs in a PE are scaled down based on the maximum VNF delay of the respective PE using Eqs. (1), (2), and (3) (step 6). The new set of PSFCs after scaling is stored in set M'. For deploying each PSFC request, the algorithm works in three stages: *PE deployment*, *PE routing*, and *VNF backup deployment*. The best PNs for deploying PEs of a PSFC are computed in the first stage (step 8). The second stage aims to find the best path from source to destination by connecting selected PNs for PEs while respecting end-to-end delay and bandwidth constraints (step 9). If the reliability requirement of PSFC is not satisfied with the primary VNF deployment, then the third stage uses a backup mechanism to ensure PSFC's reliability requirement (step 10 – step 16). The following subsections describe in detail each of these three stages.

Algorithm 1: Algorithm *ERASE*

```
1  Input: $G(V, E)$, $M$, VNF resource-delay dependency table
2  Output: PSFC placement
3  begin
4      Sort PSFCs in $M$ in non-decreasing order based on total
         resources required
5      $M' \leftarrow \{\}$
6      $M' \leftarrow$ Adjust the resources to VNFs in a PE based on
         its peak delay using Eqs. (1), (2), and (3)
7      for each PSFC request from $M'$ do
           // Stage 1: PE deployment
8          Construct the candidate PN matrix, which consists
             of a set of candidate PNs for each PE, and then
             select the suitable PN for each PE by calling
             $PE\_deployment()$    (Refer Algorithm 2)
           // Stage 2: PE routing
9          Find the route by connecting the chosen PNs in
             stage 1 while ensuring end-to-end delay and
             bandwidth constraints by calling $PE\_routing()$
             (Refer Algorithm 3)
           // Stage 3: Guaranteeing reliability
10         Evaluate the reliability of PSFC into $R_{psfc}$ using
             Eq. 13
11         if $R_{psfc} >= R_i^{req}$ then
               // no need to provide backup
12             Deploy primary VNFs of PEs on the chosen
                 PNs and update available network resources
13         end
14         else
15             $PSFC\_backup\_deployment()$    (Refer
                 Algorithm 4)
16         end
17     end
18 end
```

Algorithm 2: Procedure PE_deployment()

```
1  begin
       // Step 1: Candidate PN matrix construction
2      for each PE of PSFC request do
3          Rank the PNs based on Eq. 45 and sort based on it
4          if no PN is found for a PE which has more than one VNF then
5              $PSFC_{new} \leftarrow partial\_parallelism()$
6              Build candidate PN matrix for $PSFC_{new}$ by calling $PE\_deployment()$
7          end
8          if no PN is found for a PE which has one VNF then
9              Reject the request
10             return
11         end
12     end
       // Step 2: PN selection
13     for each PE of PSFC request do
14         Select PN which satisfies the resource requirement from the corresponding PE row in the candidate PN matrix
15         if no PN satisfies resource requirements then
16             if $(|PE_{VNF}| > 1)$ then
17                 $PSFC_{new} \leftarrow partial\_parallelism()$
18                 Build candidate PN matrix for $PSFC_{new}$ by calling $PE\_deployment()$
19             end
20             else
21                 Reject the request
22                 return
23             end
24         end
25     end
26 end
```

A. Stage1: PE Deployment

This stage's primary goal is mapping each PE of the requested PSFC to a PN in the network to reduce energy and increase reliability. PE deployment comprises three steps, namely *PN ranking*, *candidate PN matrix construction*, and *PN selection*.

1) PN ranking

We determine the set of PNs that are the best candidates for hosting P_i of PSFC i. For each PE, we rank the PNs based on the scores calculated using energy and reliability factors. We define the score of a PN $k \in N$ to host $j \in P_i$ as

$$\delta_k^j = W.\alpha_k + (1 - W).\beta_k^j \tag{43}$$

where α_k and β_k^j are reliability impact and energy impact of PN k, respectively, and W is a weight parameter representing the relative importance of energy and reliability. It is important to note that δ_k^j considers the node's current energy consumption and reliability. The weights of the α_k and β_k^j metrics can be changed according to the operator preferences/priorities.

Reliability Impact (α_k): α_k represents the reliability of a PN k, and its value is between 0 and 1. This metric allows for selecting PNs with higher reliability, hence reducing the number of backups required to achieve the PSFC reliability requirement.

Energy Impact (β_k^j): This metric aims to prioritize PNs that result in a minimum increase in the energy consumption to process a PSFC request i (either by turning on a PN which is in standby mode or using an already running one). It can take two different values based on the state of a PN.

- *β_k^j = 1:* If PN k is already running with enough available capacity to host the PE.
- *β_k^j = 0.9:* If PN k is in standby mode and must be turned on to deploy PE.

2) Candidate PN matrix construction

After assigning ranks to PNs based on energy and reliability factors, we construct a candidate matrix by arranging PNs in non-increasing order of ranks for each PE. Since all VNFs of a PE are deployed on the same PN in order to minimize additional overhead due to VNF parallelization, rows in the matrix correspond to PEs rather than VNFs. In case of conflict (same δ_k^j value for multiple PNs), we break the ties using the Available Residual Resources (ARR) policy and prioritize the PN with the higher ARR. We use γ_k for representing ARR value of a PN k, which ranges between 0 and 1 and it is calculated as

$$\gamma_k = \frac{resources\ utilized}{total\ resources} \tag{44}$$

It is important to note that we also utilize γ_k to identify candidate PNs for PEs with multiple VNFs. Since we use an onsite backup strategy for such PEs, it helps to select PNs with more ARR; backups (if required) can be made on the same PN where primary VNF is deployed, preventing conversion to partial parallelism. Therefore, the updated ranking policy used for PEs with multiple VNFs is as follows:

$$\delta_k^j = W1.\alpha_k + W2.\beta_k^j + (1 - W1 - W2).\gamma_k \tag{45}$$

$W1$ and $W2$ are weight parameters representing the relative importance of energy, reliability, and ARR. We normalize these three terms to make them belong to the same scale.

3) PN selection

After candidate PN matrix construction, we select PNs for PEs. First, we start with PEs with more than one VNF followed by PEs with a single VNF. For each PE, assign a PN with the highest rank. If the PN does not meet the required capacity, select the next PN in the candidate matrix's order. For PEs with multiple VNFs, we convert the PSFC request from full parallelism to partial parallelism (using *partial_parallelism()* procedure) if no PN is identified or candidate PNs do not meet the resource requirement. The request must be rejected for PEs with a single VNF if no PN meets resource requirements.

Partial VNF Parallelism Transition (*partial_parallelism()* procedure): We deploy all VNFs of a PE on the same PN to avoid VNF parallelization overheads. If no node is available for deploying such PEs, the PSFC request is rejected. To address this, we transition to partial parallelism, which involves dividing one PE into multiple PEs to decrease the number of parallel VNFs in each PE. It reduces the required resources for resultant PEs and increases the likelihood of finding suitable nodes. The overall latency of the PSFC may increase as a result of running some VNFs sequentially after the transition. To get all the possible sequential derived PEs of a PE of length N, we use a dynamic programming algorithm proposed in [18]. We define S_k as the set of possible sequential derived PEs for a PE of size N. For instance, for a PE of length 3, three VNFs can be executed sequentially (s = [1,

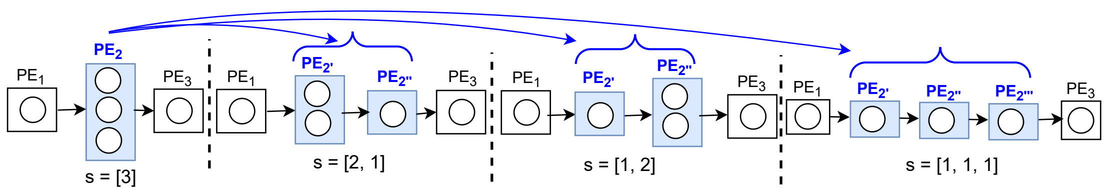


Figure 6: Possible partial parallelism transitions. Each circle represents a VNF and square represents a PE.

1, 1]), fully in parallel (s = [3]), or partially in parallel (s = [1, 2] or s = [2, 1]). Hence, we get S_3 = {[1, 1, 1], [1, 2], [2, 1], [3]}. Among these, we prefer the combination that gives maximum latency benefits. It can be accomplished by arranging VNFs in decreasing order based on their processing delays and assigning them to various combinations. Fig. 6 demonstrates three potential partial parallelism combinations for a PE of size three.

Algorithm 2 describes the PE deployment process of a given PSFC. Step 2 – step 12 explain the procedure for constructing the candidate PN matrix of a given PSFC, and step 13 – step 25 describe the PN selection procedure for PEs using the candidate matrix.

B. Stage2: PE Routing

Given a list of selected PNs for PEs, *PE routing* computes the path from source to destination that traverses the selected PNs and satisfies the delay and bandwidth requirements. Instead of running Dijkstra's algorithm directly from the source to the destination node, we apply the Dijkstra's algorithm sequentially from the source node to the node that hosts first PE, followed by the node that hosts the second PE, and so on until the destination. Hence, our algorithm ensures that it only considers paths that traverse the list of selected PNs and the final path satisfies the delay and bandwidth constraints. The working of PE routing is given in Algorithm 3.

If the end-to-end delay requirement is satisfied, we attempt to reduce energy consumption by altering the path between PNs. Note that the energy consumption of a physical link relies on its on/off state and bandwidth utilization. We determine the energy consumption of the links between sub-paths of the selected PNs and then order the sub-paths depending on their energy consumption. For each sub-path, starting with the one that consumes the most energy, we compute the k shortest paths (using Yen's algorithm [41]) between the corresponding end-points of the sub-path and select the one that consumes least energy while meeting the delay and bandwidth requirements. This procedure is repeated for the end points of every sub-path. It returns a lowest energy path that does not exceed the delay requirement (step 4 – step 10).

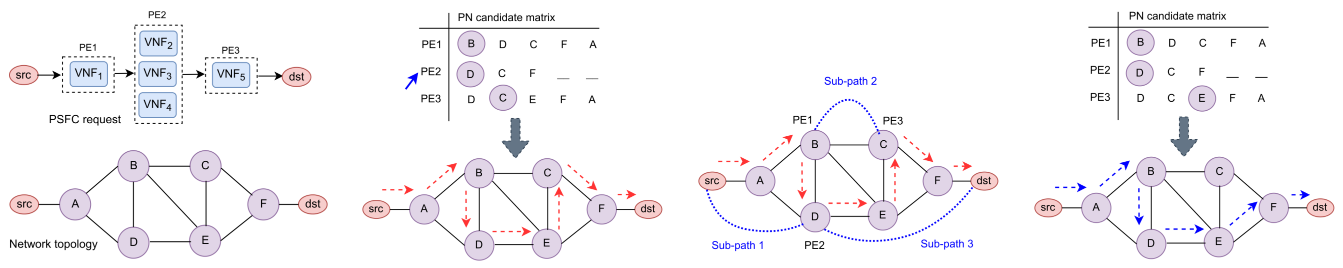
If the end-to-end delay constraint is not met, we consider redeployment for certain PEs, choose alternative PNs from the candidate PN matrix, and repeat the PE routing procedure. Since different PNs for deploying PEs lead to different solutions, we need to determine an alternative list of selected PNs for PE deployment which is expected to reduce energy consumption while allowing a path passing through them that guarantees the constraints. Having the PN candidate matrix in hand, we present a mechanism, namely *Largest Sub-path Delay (LSD)*, PN selection mechanism to determine the alternative list of selected PNs for PE deployment (step 12).

Largest Sub-path Delay (LSD): This mechanism uses a sub-path delay metric to eliminate a PN from the selected PN list. We first partition the shortest path from s_i to d_i passing through PNs (wherein PEs are deployed) into e sub-paths: s_i to PE_2 (PE_1 is the intermediate PE), PE_1 to PE_3, and PE_{e-1} to d_i. Note that each sub-path comprises a single intermediary PN that hosts a PE. For each sub-path, the shortest path between end points of the sub-path is calculated using the delay as the cost function. Then, the intermediate PN of the sub-path with the longest sub-path delay is eliminated from the current PN list. Consequently, the eliminated PN is replaced with the next candidate PN in the corresponding row of the candidate PN matrix. This process iterates until either a solution meeting the delay and capacity constraints is discovered or the stopping condition is met. The stopping condition signifies that all possible PNs for deploying the corresponding intermediate PEs have been explored. It indicates that the deployment process has exhaustively considered the available options and has reached the end of the exploration phase.

For better understanding, we illustrate the LSD process by taking an example. Fig. 7a shows the considered network topology and PSFC request. The candidate PN matrix and initial PN selection corresponding to the PSFC request are shown in Fig. 7b. Assume that end-to-end delay is not satisfying with the initial PN selection for PEs of the given SFC in Fig. 7b. Fig. 7c shows three different sub-paths for the considered PSFC, whereas Fig. 7d shows an example of the redeployment phase (chosen sub-path 3) in which it decides to replace the PN C where PE_3 is deployed with PN E and selects the corresponding path from the source to the destination via newly chosen node.

C. Stage3: VNF Backup Deployment

Until stage 2, PEs are assigned to PNs in a way that minimizes energy consumption and improve reliability while meeting resource and delay requirements. However, it may or may not meet the PSFC reliability requirements. The reliability of a PSFC is computed using Eq. 13. The conventional approach to increase the reliability of a PSFC is to create redundant backups for VNFs in PEs. However, backup models introduce additional resource consumption. Typically, 1:1 redundancy architecture has been proven ineffective [24], [42], [43]. In this work, we investigate the minimum number of backup VNFs for the whole PSFC request to be created by the service provider to guarantee a certain degree of reliability. Service providers need a dedicated mechanism to ensure reliability at relatively low resource consumption, overhead, and delay.



(a) PSFC request and topology (b) Initial deployment (c) Different sub-paths (d) Redeployment

Figure 7: A high-level example that illustrates the steps that the *ERASE* takes to find a solution.

Algorithm 3: Procedure PE_routing()

```
1  begin
2      For each pair of selected PNs compute the shortest path
         while ensuring bandwidth requirement
3      delay ← compute the end-to-end delay
4      if $delay < D_i^{req}$ then
5          for each sub-path do
6              Compute the $k$ shortest paths between the
                 end-points
7              Select the path that consumes lesser energy than
                 current energy while meeting the delay and
                 bandwidth requirements
8          end
9          return
10     end
11     else
12         Follow the LSD mechanism for choosing different
             PNs and find the route by calling $PE\_routing()$
13     end
14 end
```

Algorithm 4 describes the process of backup VNF deployment. It iteratively adds backup VNFs until the required reliability requirements are met or reach the maximum number of backup VNFs. We allow at most one backup for each VNF. Creating backup for the most unreliable VNF will increase the reliability significantly. Thus, we sort VNFs based on their reliability values and store them in a list $S^{'}$(step 2). The backup VNF instances are preferentially placed on PNs where their primary VNFs are located. This way, the overhead of synchronizing the states of the primary VNF and backup VNF can be reduced. If the PE has a single VNF, then we give the first preference to the onsite backup policy. If it is not possible to deploy the backup and primary PE on the same PN, then we follow the offsite backup policy and choose a node for creating a backup and reserving computing resources. In contrast to the traditional approach, the offsite backup policy places backup VNFs further downstream/upstream along the PSFC's path so that it always meets the end-to-end delay and bandwidth requirements (step 4 – step 11). It is important to note that there is no need to reserve extra bandwidth resources, which may be utilized to deploy more number of PSFC requests.

If the PE has multiple VNFs, we always follow the onsite backup policy for creating a backup instance to minimize the overheads introduced by VNF parallelization. If the PN where such PEs are deployed has insufficient capacity to place a backup VNF instance, we avoid taking backup for that particular VNF and continue with other VNFs (step 12 – step 19). It is important to note that in cases where taking

Algorithm 4: Procedure PSFC_backup_deployment()

```
1  begin
2      Sort VNFs in non-decreasing order based on their
         reliability into list $S^{'}$
3      for each VNF from $S^{'}$ do
4          if $(|PE| = 1)$ then
5              if PN has sufficient capacity then
6                  Mark this PN into temp    // onsite backup
                     strategy
7              end
8              else
                   // offsite backup strategy
9                  Find PN either downstream or upstream
                     along the PSFC's path and mark into temp
10             end
11         end
12         else
13             if PN has sufficient capacity then
14                 Mark this PN into temp
15             end
16             else
17                 continue
18             end
19         end
20         Find the updated reliability using Eq. 13 into $R^{psfc}$
21         if $R^{psfc} >= R^{req}$ then
22             Deploy the backup VNF into temp PN and
                 update the available network resources
23             return
24         end
25     end
26     return
27 end
```

backup for such VNF leads to satisfying PSFC reliability, we reject the request as it leads to additional VNF parallelization overheads due to the deployment of backup VNF instance in another node. We check whether reliability is guaranteed or not after creating every backup and repeat this procedure for all VNFs until the reliability requirements have been met. If the desired reliability is satisfied, then we deploy backup VNF instances and update the available network resources (step 20 – step 24).

D. Complexity Analysis

Let F represent the maximum number of VNFs in any PSFC request. Algorithm 1 employs three stages for deploying a PSFC request. In the first stage, we construct a candidate PN matrix. Rows and columns in the matrix correspond to PEs in the PSFC and candidate PNs, respectively. In the worst case, the set of PEs in a PSFC is same as set of

VNFs. For each PE, we arrange candidate PNs in ascending order of scores calculated using energy and reliability factors. Thus, the time complexity for creating a candidate PN matrix is $O(F|N|\log|N|)$. After this matrix construction, we select PNs for deploying PEs which can take $O(F|N|)$ time in the worst case. Hence the total time complexity of stage 1 is $O(F|N|\log|N|)$. Then, we iteratively compute the route for connecting the chosen PNs by computing k shortest paths between end points of each sub path. The number of PNs is $F+2$ including source and destination nodes, thus, we need to find k shortest paths between $F+1$ subpaths. Hence, the overall time complexity of stage 2 is $O(kF|N|^2)$. In general, k can be ignored as it is constant and a small number. In stage 3, we sort the VNFs in non-decreasing order of their reliability and create backups until meeting the reliability requirement. Since the complexity of the VNF backup deployment algorithm only depends on the number of PNs in the path between the source and destination, it does not impact much compared to the first two stages. These three stages are repeated for each PSFC deployment. Thus, the time complexity of *ERASE* is $O(MF|N|^2)$).

VII. Performance Evaluation

In this section, we first present the experimental setup used for performance evaluation. Then we report the performance of our proposed scheme *ERASE* and compare it with some baseline and bechmark schemes.

A. Simulation Setup

To evaluate the performance of the proposed *ERASE* scheme, we developed a C++ based simulator. We evaluated *ERASE* scheme on a medium-scale network topology (USNET with 24 nodes and 43 links) and a large-scale network topology (CORONET with 75 nodes, 99 links). The simulation parameters are chosen from [3], [11], [17], [44]–[47], which are given in Table III. Unless otherwise stated, these parameters are used as the default settings. VNFs of each PSFC are picked from 10 different VNFs, and the source and destination pair of each PSFC is picked randomly from the underlying network topology. Fig. 8 illustrates the effect of the weight factor W on the energy consumption and reliability achieved for 50 PSFC requests. As the value of W increases, the average total energy consumption increases as more importance is given to reliability than energy. It can be observed that when the value of W is very high, for example, 0.9, energy consumption is higher with high average achieved reliability and vice versa for low W values. Based on this result, we set W as 0.4 for further experiments as it balances the relative importance between energy and reliability. In all our experiments, we set $W1$ and $W2$ as 0.4 and 0.5, respectively, and the weight of ARR is set to 0.1. To minimize the impact of random factors on the results, each simulation experiment is repeated 50 times, and the graphs are plotted with a 95% confidence interval.

B. Benchmark Schemes

Since this paper is the very first to study the energy-aware reliable placement of parallelized SFCs in NFV-based systems while guaranteeing delay requirements, we present two benchmark schemes to investigate the performance of the proposed scheme *ERASE*.

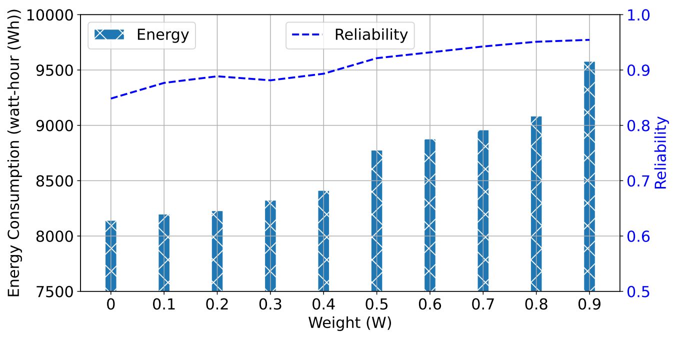


Figure 8: Impact of W on energy and reliability.

- **Energy-aware only scheme (*EoS*):** The main difference between this scheme and *ERASE* is that it selects PNs for deploying PEs only based on energy factor (i.e., $W1$ and $W2$ are set as 0 and 0.9, respectively). It indicates that it prioritizes PNs based on their energy consumption regardless of their reliability.
- **Reliability-aware only scheme (*RoS*):** The main difference between this scheme and ERASE is that it only considers the reliability factor when selecting PNs for deploying PEs (i.e., $W1$ and $W2$ are set as 0.9 and 0, respectively). It indicates that it prioritizes PNs based on their reliability regardless of their energy consumption.

To demonstrate the impact of SFC parallelization, we have also implemented the following three variants of sequential SFC placement to provide a thorough evaluation.

- Energy-aware efficient sequential SFC placement (*ESP*)
- Reliability-aware efficient sequential SFC placement (*RSP*)
- Energy- and Reliability-aware efficient sequential SFC placement (*ERSP*)

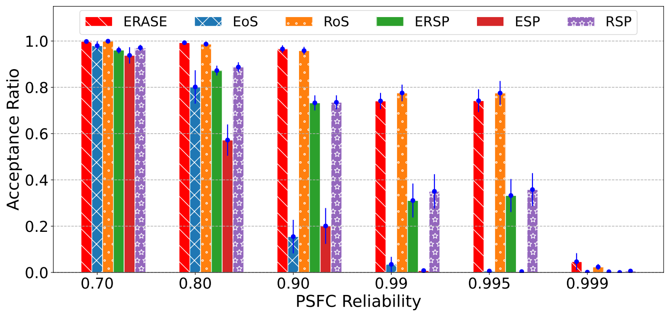
It is worth mentioning that the schemes used for sequential SFC deployment, such as *ESP*, *RSP*, and *ERSP*, have similar functionality to those employed in parallelized SFC deployment, namely *EoS*, *RoS*, and *ERASE*. *ESP*, *RSP*, and *ERSP* are designed to operate on sequential SFCs, whereas *EoS*, *RoS*, and *ERASE* are tailored for parallelized SFCs.

The following performance metrics are used to evaluate the performance of the schemes mentioned above.

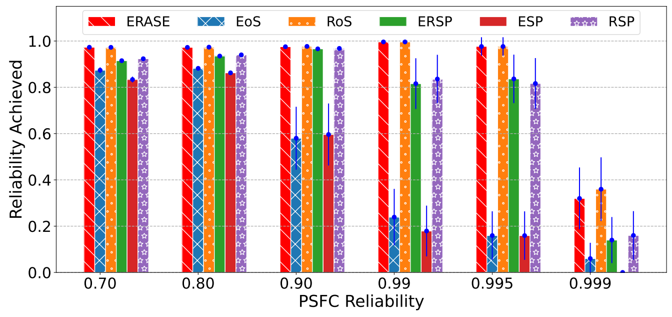
1) **Acceptance ratio:** It is the ratio between the number of PSFC requests that were accepted in the network and the total number of requests. An PSFC request is said to be accepted when its end-to-end delay and reliability requirements are met.
2) **Average end-to-end delay:** It represents the average end-to-end delay (i.e., the sum of processing delay and propagation delay) of PSFC requests that are accepted.
3) **Average reliability:** The average reliability of PSFC requests that are accepted.
4) **Average number of backup VNFs:** Average number of backup VNF instances created for meeting the reliability requirements of accepted PSFC requests.

Table III: Simulation parameters

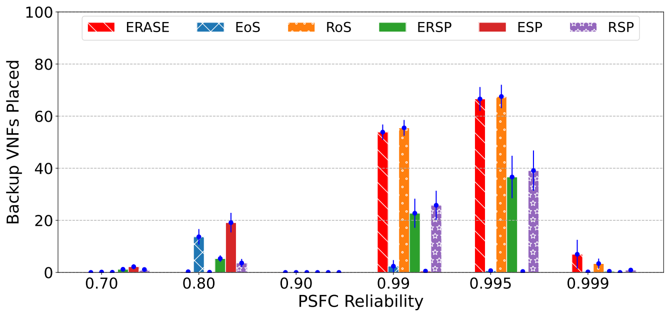
PSFC Requirements		Physical Network	
Parameters	**Values**	**Parameters**	**Values**
Length of PSFC requests	[4 - 8]	Node resource capacity	[250 - 450] units
VNF processing delay	[5 - 10] msec	Link capacity	[5 - 10] Gbps
VNF resource request	[5 - 10] units / Gbps	Link delay	[4 - 7] msec
Traffic rate	[100 - 300] Mbps	Reliability of each physical node	[0.9, 0.99, 0.999, 0.9999]
End-to-end delay	[80 - 120] msec	Starting and peak energy of physical node	299 watt-hour (Wh), 500 watt-hour
Reliability of VNF	[0.9, 0.99, 0.999, 0.9999]	Starting and peak energy of link	50 watt-hour, 200 watt-hour
PSFC reliability	[0.75 - 0.99]	Number of PSFC requests	50 (USNET), 100 (CORONET)



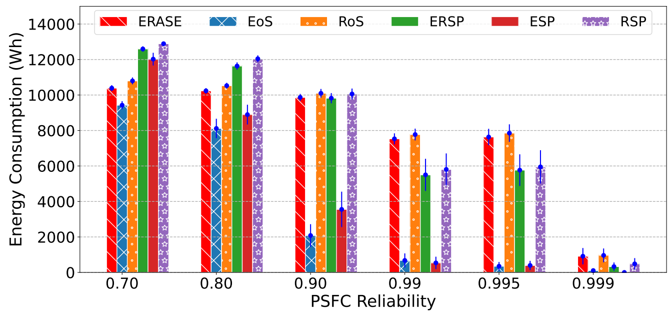
(a) Acceptance ratio



(b) Reliability achieved



(c) Backup VNFs deployed



(d) Total energy consumption

Figure 9: Simulation results w.r.t varying PSFC reliability with USNET network topology.

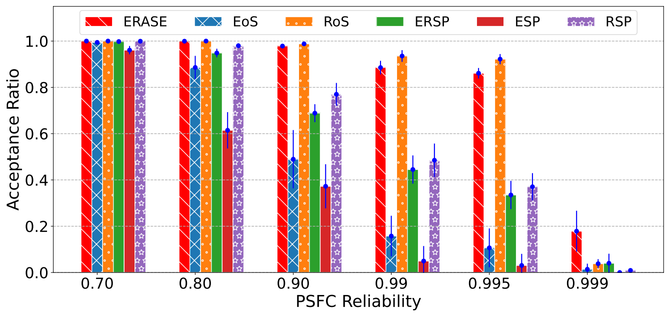
5) **Energy consumption:** The total amount of energy consumption of the network, which consists of energy consumption of all the servers and physical links of the accepted PSFC requests.
6) **Number of running servers:** The number of servers activated for running at least one VNF.

In the following subsections, we first evaluate the performance of the ERASE with *EoS* and *RoS* through extensive simulations and then we compare its performance with three variants of sequential SFC deployment, namely *ESP*, *RSP*, and *ERSP*.

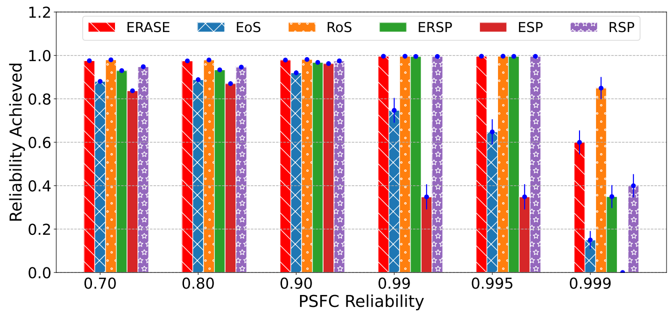
C. *Impact of the PSFC reliability*

Fig. 9 shows the performance of the proposed *ERASE* scheme along with five benchmark schemes with the variation of PSFC reliability for the USNET network topology. Figs. 9a and 9b illustrate the acceptance ratio and the corresponding average reliability achieved by all schemes, respectively. At a reliability requirement of 0.70, *ERASE EoS*, and *RoS* schemes accepted all PSFC requests, but the *EoS* scheme had an average reliability of 0.85, while *ERASE* and *RoS* schemes had an average reliability of 0.96. The *EoS* scheme does not take into account node reliability while placing VNFs, resulting in a decrease in the average PSFCs reliability. In contrast, the *ERASE* and *RoS* schemes considered node reliability factors, resulting in a high average PSFC reliability. The *EoS* scheme began rejecting requests at a reliability requirement of 0.80 or higher, while the *ERASE* and *RoS* schemes began rejecting requests at a reliability requirement of 0.95 or higher. The *ERASE* scheme performed similarly to the *RoS* scheme in terms of achieved reliability and acceptance ratio.

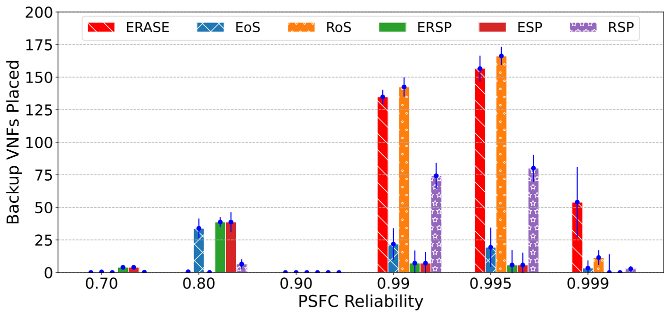
The number of backup VNFs required by all six schemes for each reliability criterion is shown in Fig. 9c. The *EoS* scheme does not require backups when reliability requirements are less than 0.80, but as reliability requirements increase, the acceptance ratio drops, and no backups are placed from a reliability requirement of 0.90 as shown in Fig. 9c. Neither *ERASE* nor *RoS* schemes require backups to accept requests with reliability requirements less than 0.99. The number of backups increases as reliability requirements increase from 0.99 to 0.995, but even with backups in place, the requirement is not met at 0.999. The energy consumed by each scheme at different reliability requirements is shown in Fig. 9d. At a reliability requirement of 0.70, *ERASE EoS*, and *RoS* schemes accept all requests. The *EoS* scheme consumes the least energy, as it prioritizes energy-related criteria in node selection. As reliability requirements increase, the *EoS* scheme consumes less energy due to its decreasing acceptance ratio. *ERASE* outperforms *RoS* in total energy consumption by an average of 10% across all reliability requirements, as it considers a combination of reliability and energy factors while placing



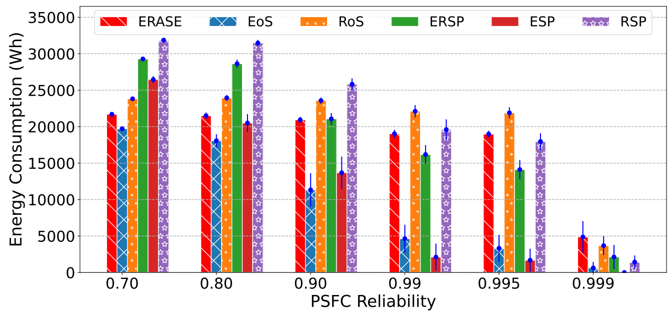
(a) Acceptance ratio



(b) Reliability achieved

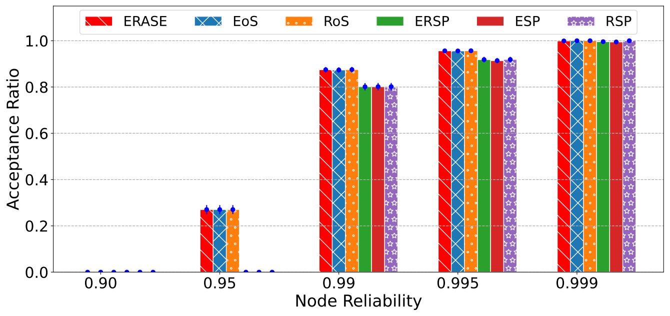


(c) Backup VNFs deployed

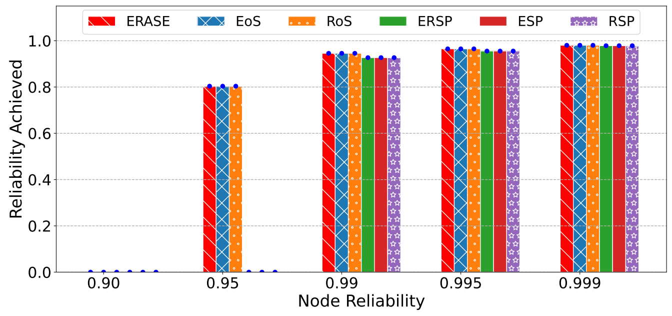


(d) Total energy consumption

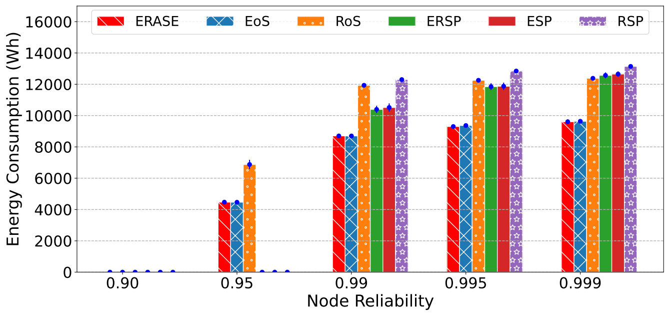
Figure 10: Simulation results w.r.t varying PSFC reliability with CORONET network topology.



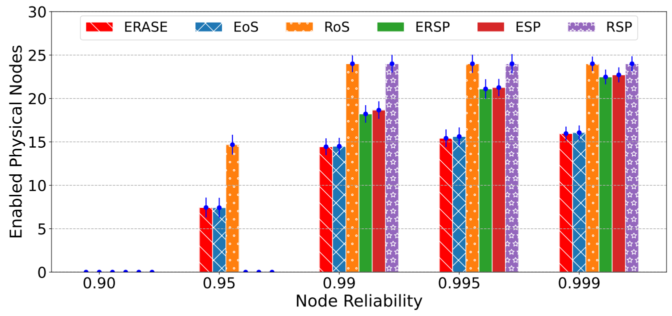
(a) Acceptance ratio



(b) Reliability achieved



(c) Total energy consumption



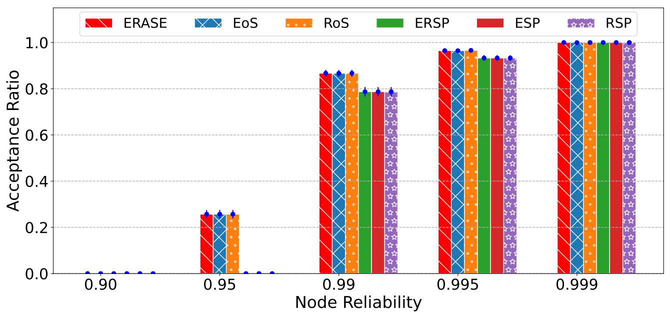
(d) Number of running servers

Figure 11: Simulation results w.r.t varying physical node reliability in USNET network topology.

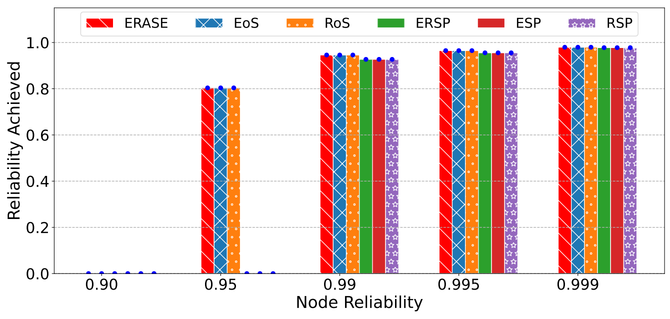
PSFC requests, thus reducing energy consumption. Overall, the proposed *ERASE* scheme has similar performance to *EoS* in terms of energy consumption and *RoS* in terms of reliability. Similar trends were also observed in the larger CORONET network topology, where *ERASE* consumed about 20% less energy than *RoS* and almost the same energy as the *EoS* scheme as shown in Fig. 10.

D. Impact of the physical node reliability

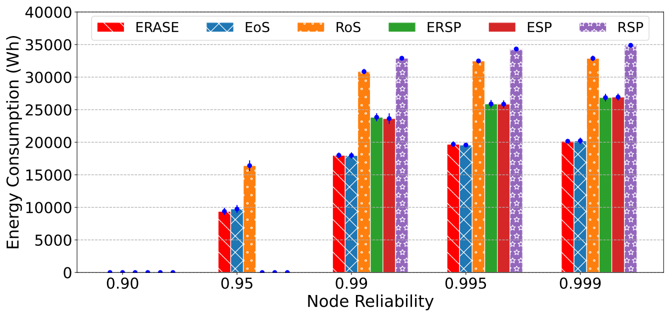
Fig. 11 depicts the performance of the proposed *ERASE* scheme along with five benchmark schemes by varying the node reliability for the USNET network topology. Figs. 11a and 11b depict the acceptance ratio and the achieved reliability, respectively. We observe that *ERASE*, *RoS*, and *EoS* schemes perform similarly. This is because all nodes have the same



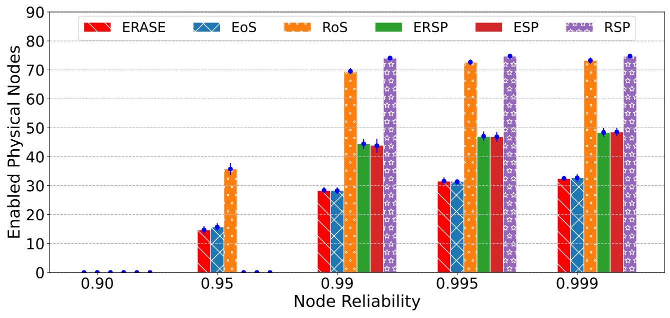
(a) Acceptance ratio



(b) Reliability achieved

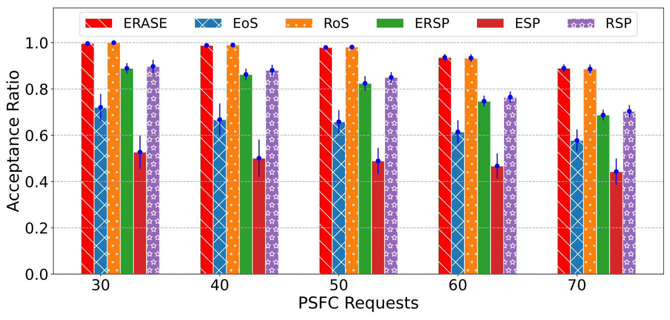


(c) Total energy consumption

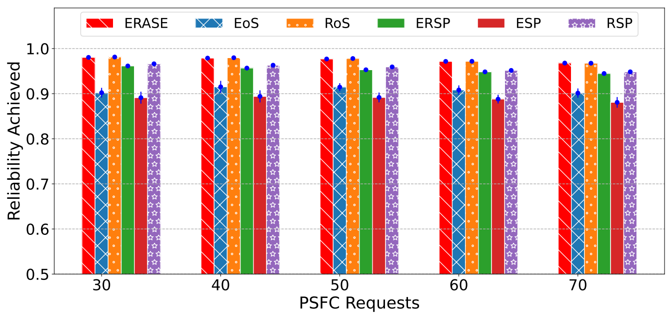


(d) Number of nodes enabled

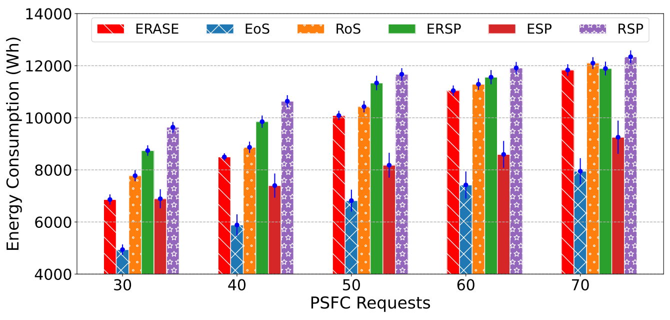
Figure 12: Simulation results w.r.t varying physical node reliability in CORONET network topology.



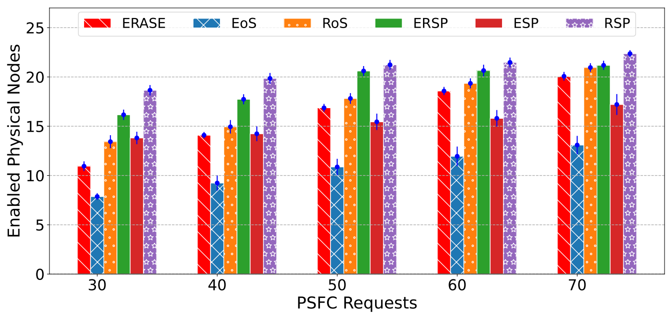
(a) Acceptance ratio



(b) Reliability achieved



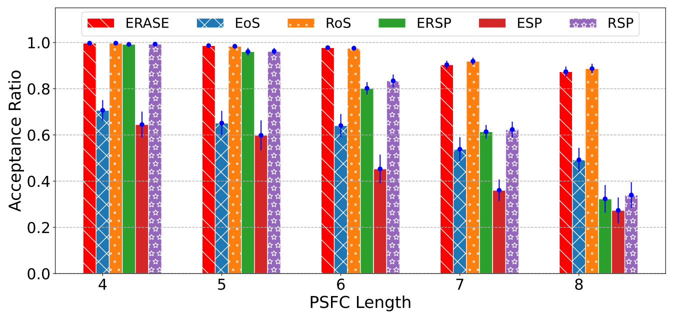
(c) Total energy consumption

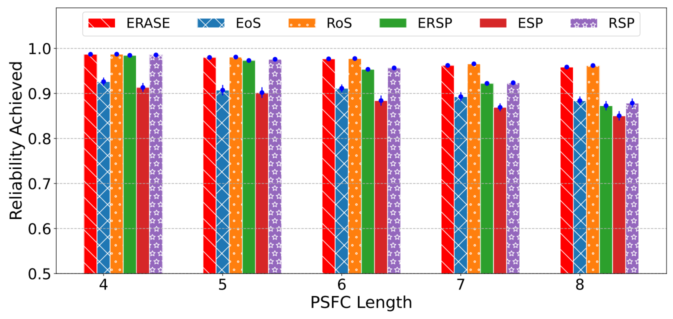


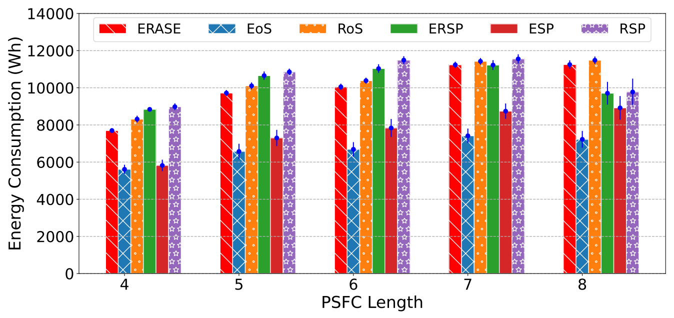
(d) Number of nodes enabled

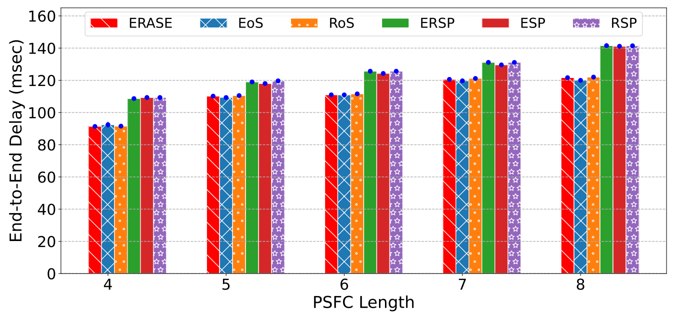
Figure 13: Simulation results w.r.t varying number of PSFC requests with USNET network topology.

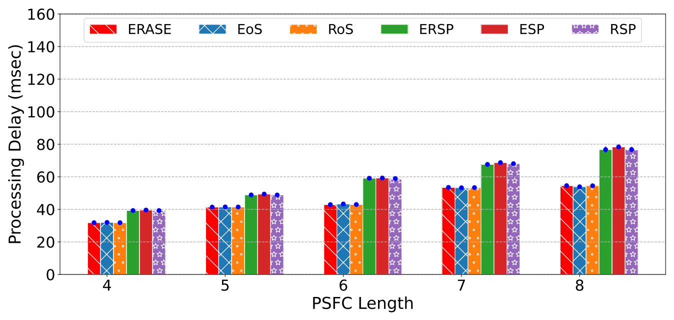
node reliability and are regarded equally in terms of reliability. No matter which nodes are chosen for VNF deployment, the final PSFC reliability will be the same. Note that *ERASE*, *RoS*, and *EoS* schemes place VNFs for a specific request on distinct nodes. Observing the trend in Fig. 11a, increased node reliability results in a rise in the acceptance ratio. This is because, as shown in Fig. 11b, the higher the node's reliability, the higher the PSFC's reliability will be, allowing it to accept more requests. Figs. 11c and 11d show the total energy consumed and the number of running servers of the proposed *ERASE* and five benchmark schemes. The *RoS* scheme results in an increase in the total energy consumption because it only considers the reliability of nodes before using them to place requests, leading to the activation of more servers. On the


(a) Acceptance ratio


(b) Reliability achieved


(c) Total energy consumption


(d) Average end-to-end delay


(e) Average VNF processing delay

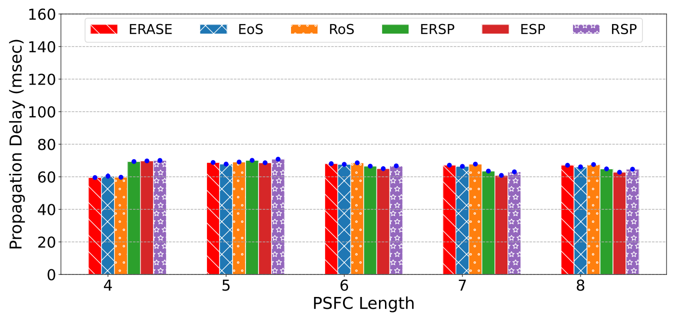

(f) Average propagation delay

Figure 14: Simulation results w.r.t varying PSFC length with USNET network topology.

other hand, the *ERASE* and *EoS* schemes provide weight to already enabled nodes, enabling fewer number of nodes and consuming less energy. Note that energy consumption is also dependent on the total number of PSFC requests accepted. We observe an upward trend in energy consumption as the reliability of a node increases. The ERASE scheme is found to consume 60% less energy than the *RoS* scheme and almost the same as the *EoS* scheme. Similar trends are observed in the larger CORONET network topology as shown in Fig. 12.

E. Impact of the number of PSFC requests

In Fig. 13, the performance of the proposed *ERASE* and five benchmark schemes is observed by varying the number of PSFC requests for the USNET network topology. Figs. 13a and 13b show the acceptance ratio and reliability achieved for varying number of PSFC requests. In the *EoS* scheme, it is seen that the average reliability achieved is almost the same regardless of the number of requests, this is because the *EoS* scheme does not consider reliability parameters before deploying VNFs. It is observed that the rejection of requests is mainly due to the depletion of node resources rather than the request reliability requirement. As the number of requests increases, the acceptance ratios of *ERASE*, *EOS*, and *ROS* schemes decrease. Due to the requirement for PSFC request reliability, the *EoS* scheme has a lower acceptance ratio than the *ERASE* and *ROS* schemes. As more and more requests are accepted, the high-reliability nodes become depleted, which leads to a reduction in the average reliability achieved by *RoS* and *ERASE* schemes.

Figs.13c and 13d depict the total amount of energy consumed and the number of servers enabled by all six schemes. As expected, *RoS* consumes more energy as it enables more servers. The *EoS* scheme consumes significantly less energy than *ERASE* due to the fact that its acceptance ratio is 10 to 15 percent lower than *ERASE* over all iterations, thereby using less energy. We also observed similar trends for larger CORONET network topology.

F. Impact of the PSFC length

In Fig. 14, the performance of the proposed *ERASE* and five benchmark schemes is observed by varying the PSFC length for the USNET network topology. Fig. 14a and 14b show the acceptance ratio and the corresponding average reliability achieved by all schemes, respectively. With increasing PSFC length, the achieved reliability decreases, and this trend is observed in *ERASE*, *EoS*, and *RoS* schemes. The *EoS* scheme gives the lowest reliability for all PSFC lengths because it chooses nodes for placing VNFs without considering reliability factors. The acceptance ratio depends on the required level of request reliability. Since *ERASE* and *RoS* schemes achieve comparable levels of reliability, they also produce comparable acceptance ratios. Since the achieved request reliability decreases as the length of the PSFC chain increases, the acceptance ratio of *ERASE* and *RoS* schemes also decreases.

The energy consumed by the proposed *ERASE* and five benchmark schemes is depicted in Fig. 14c. Energy consumption is dependent on the acceptance ratio and the length of the PSFC request. As the number of VNFs in a PSFC increases, more energy is consumed as more nodes are required to place them. Therefore, *ERASE*, *EoS*, and *RoS* schemes consume more energy as the length of the chain increases. The *EoS* scheme has a lower acceptance ratio and thus uses less energy than *ERASE* and *RoS* schemes. Compared to the *RoS* scheme, *ERASE* saves a significant amount of energy while maintaining the same acceptance ratio, as it takes energy parameters also into account when placing VNFs. Fig. 14d depicts the average end-to-end delay of requests of different PSFC lengths. As the length of PSFC increases the total end-to-end delay increases. This is mainly because of the larger number of VNFs in the PSFC, resulting in additional processing delays. Fig. 14e and Fig. 14f show the impact of VNF processing delay and propagation delay on the total end-to-end delay, respectively. Similar trends were observed in the larger CORONET network topology.

G. Sequential vs. Parallelized SFC Deployment

The results for both sequential and parallelized SFC deployments are presented in Figs. 9 to 14. In terms of various aspects, such as PSFC reliability, physical node reliability, number of PSFC requests, and PSFC length, the parallelized SFC schemes consistently outperform different variations of sequential SFC deployment schemes across all performance metrics, including reliability, energy efficiency, end-to-end delay, and acceptance ratio. This superiority can be attributed to three significant factors. Firstly, the implementation of flexible resource allocation for parallelized VNFs enables efficient resource utilization, leading to resource savings and the capability to accommodate a higher number of requests. Secondly, the parallel execution of specific VNFs effectively reduces latency, resulting in enhanced overall performance. Lastly, the deployment of sequential SFC requests across multiple nodes introduces a trade-off in terms of reliability, which can potentially impact the overall reliability of the SFC.

VIII. Conclusions and Future Work

This paper introduced a novel parallelized SFC deployment problem with the objective of reducing energy consumption while meeting reliability and delay constraints. First, the problem was formulated as an ILP model, and then, due to its NP-hardness, a heuristic solution called *ERASE* was proposed that consists of three stages: PE deployment, PE routing, and VNF backup deployment. Extensive simulation results showed that the proposed solution can significantly reduce the total energy consumption in comparison to benchmark schemes while guaranteeing the delay and reliability requirements of PSFC requests.

Some potential future directions of this work are as follows. In this work, all parallelizable VNFs were deployed on the same server to avoid any additional overheads. We plan to evaluate the impact of overheads such as packet deposition and packet duplication/merging when parallelizable VNFs are deployed on different servers. Providing online algorithms for deploying PSFC requests with the objective of minimizing the energy while considering the parallelization overheads, and meeting the delay and reliability requirements is a very important problem to investigate. Reinforcement learning could play a big role in addressing this problem.

References

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

[2] N. ISG, “Network functions virtualisation (nfv)-virtualisation requirements,” *ETSI Technical Report*, 2013.

[3] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca, “Holu: power-aware and delay-constrained vnf placement and chaining,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1524–1539, 2021.

[4] R. Lin, L. He, S. Luo, and M. Zukerman, “Energy-aware service function chaining embedding in nfv networks,” *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1158–1171, 2022.

[5] M. Chen, Y. Sun, H. Hu, L. Tang, and B. Fan, “Energy-saving and resource-efficient algorithm for virtual network function placement with network scaling,” *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 29–40, 2021.

[6] J. Liu and Q. Zhang, “Offloading schemes in mobile edge computing for ultra-reliable low latency communications,” *IEEE Access*, vol. 6, pp. 12 825–12 837, 2018.

[7] Sun, Chen and Bi, Jun and Zheng, Zhilong and Yu, Heng and Hu, Hongxin, “Nfp: Enabling network function parallelism in nfv,” in *Proc. of ACM SIGCOMM*, 2017.

[8] Y. Zhang, B. Anwer, V. Gopalakrishnan, B. Han, J. Reich, A. Shaikh, and Z.-L. Zhang, “Parabox: Exploiting parallelism for virtual network functions in service chaining,” in *Proc. of the Symposium on SDN Research (SOSR)*, 2017.

[9] L. Mai, Y. Ding, X. Zhang, L. Fan, S. Yu, and Z. Xu, “Energy efficiency with service availability guarantee for network function virtualization,” *Elsevier Future Generation Computer Systems*, vol. 119, pp. 140–153, 2021.

[10] M. M. Tajiki, M. Shojafar, B. Akbari, S. Salsano, M. Conti, and M. Singhal, “Joint failure recovery, fault prevention, and energy-efficient resource management for real-time sfc in fog-supported sdn,” *Elsevier Computer Networks*, vol. 162, p. 106850, 2019.

[11] S. Agarwal, V. R. Chintapalli, and B. R. Tamma, “Flexsfc: Flexible resource allocation and vnf parallelism for improved sfc placement,” in *Proc. of IEEE NetSoft*, 2022.

[12] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, “Delay-aware vnf placement and chaining based on a flexible resource allocation approach,” in *Proc. of IEEE CNSM*, 2017.

[13] S. Xie, J. Ma, and J. Zhao, “Flexchain: Bridging parallelism and placement for service function chains,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 195–208, 2021.

[14] J. Luo, J. Li, L. Jiao, and J. Cai, “On the effective parallelization and near-optimal deployment of service function chains,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1238–1255, 2021.

[15] J. Cai, Z. Huang, J. Luo, Y. Liu, H. Zhao, and L. Liao, "Composing and deploying parallelized service function chains," *Journal of Network and Computer Applications*, vol. 163, p. 102637, 2020.

[16] K. C.-J. Lin and P.-L. Chou, "Vnf embedding and assignment for network function parallelism," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1006–1016, 2022.

[17] J. Cai, Z. Huang, L. Liao, J. Luo, and W.-X. Liu, "Appm: adaptive parallel processing mechanism for service function chains," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1540–1555, 2021.

[18] I.-C. Lin, Y.-H. Yeh, and K. C.-J. Lin, "Toward optimal partial parallelization for service function chaining," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2033–2044, 2021.

[19] H. Yu, T. Taleb, and J. Zhang, "Deterministic service function chaining over beyond 5g edge fabric," in *Proc. of IEEE GLOBECOM*, 2021.

[20] ——, "Deterministic latency/jitter-aware service function chaining over beyond 5g edge fabric," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2148–2162, 2022.

[21] Q. Zhang, F. Liu, and C. Zeng, "Online adaptive interference-aware vnf deployment and migration for 5g network slice," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2115–2128, 2021.

[22] M. Blöcher, R. Khalili, L. Wang, and P. Eugster, "Letting off steam: Distributed runtime traffic scheduling for service function chaining," in *Proc. of IEEE INFOCOM*, 2020.

[23] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "Grep: Guaranteeing reliability with enhanced protection in nfv," in *Proc. of ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, 2015.

[24] L. Yang, J. Jia, H. Lin, and J. Cao, "Reliable dynamic service chain scheduling in 5g networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 08, pp. 4898–4911, 2023.

[25] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. of IEEE INFOCOM*, 2017.

[26] Y. Wang, L. Zhang, P. Yu, K. Chen, X. Qiu, L. Meng, M. Kadoch, and M. Cheriet, "Reliability-oriented and resource-efficient service function chain construction and backup," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 240–257, 2020.

[27] D. Zhai, X. Meng, Z. Yu, and X. Han, "Reliability-aware service function chain backup protection method," *IEEE Access*, vol. 9, pp. 14 660–14 676, 2021.

[28] A. Ben Hamed, A. Leivadeas, M. Falkner, and N. Pitaev, "Vnf chaining performance characterization under multi-feature and oversubscription using sr-iov," *Informatics*, vol. 7, no. 3, 2020.

[29] S. N. Tumchou, A. Leivadeas, M. Falkner, and N. Pitaev, "Impact of cpu and memory resource allocation in a multi-feature vnf deployment," in *Proc. of IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021.

[30] V. Jain, H.-T. Chu, S. Qi, C.-A. Lee, H.-C. Chang, C.-Y. Hsieh, K. Ramakrishnan, and J.-C. Chen, "L25gc: A low latency 5g core network based on high-performance nfv platforms," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022.

[31] D. Zheng, G. Shen, X. Cao, and B. Mukherjee, "Towards optimal parallelism-aware service chaining and embedding," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2063–2077, 2022.

[32] X. Lin, D. Guo, Y. Shen, G. Tang, B. Ren, and M. Xu, "Sft-box: An online approach for minimizing the embedding cost of multiple hybrid sfcs," *IEEE/ACM Transactions on Networking*, (Early Access).

[33] J. Zhou, G. Feng, and Y. Gao, "Network function parallelization for high reliability and low latency services," *IEEE Access*, vol. 8, pp. 75 894–75 905, 2020.

[34] M. Pedram and I. Hwang, "Power and performance modeling in a virtualized server system," in *2010 39th International Conference on Parallel Processing Workshops*. IEEE, 2010, pp. 520–526.

[35] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *Proc. of IEEE INFOCOM*, 2014.

[36] N. Isg, "Network functions virtualisation (nfv); reliability; report on models and features for end-to-end reliability," *ETSI GS NFV-REL*, vol. 1, p. v1, 2016.

[37] D. Li, P. Hong, K. Xue, and J. Pei, "Availability aware vnf deployment in datacenter through shared redundancy and multi-tenancy," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[38] L. Chen, S. Patel, H. Shen, and Z. Zhou, "Profiling and understanding virtualization overhead in cloud," in *Proc. of IEEE International conference on parallel processing*, 2015, pp. 31–40.

[39] D. Li, P. Hong, K. Xue, and j. Pei, "Virtual network function placement considering resource optimization and sfc requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1664–1677, 2018.

[40] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. of IEEE INFOCOM*, 2015.

[41] E. Q. Martins and M. Pascoal, "A new implementation of yen's ranking loopless paths algorithm," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, no. 2, pp. 121–133, 2003.

[42] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proc. of the ACM SIGCOMM*, 2011.

[43] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters," in *Proc. of the conference on Internet measurement conference*, 2013.

[44] R. Gour, V. Sikand, J. Chang, Z. Wu, G. Ishigaki, and J. P. Jue, "Traffic-weighted availability-guaranteed network slice composition with vnf replications," in *Proc. of IEEE ICC*, 2022.

[45] M. Niu, Q. Han, B. Cheng, M. Wang, Z. Xu, W. Gu, S. Zhang, and J. Chen, "Hars: A high-available and resource-saving service function chain placement approach in data center networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 829–847, 2022.

[46] X. Yin, B. Cheng, M. Wang, and J. Chen, "Availability-aware service function chain placement in mobile edge computing," in *Proc. of IEEE World Congress on Services (SERVICES)*, 2020.

[47] Google., "Google apps service level agreement," ://www.google.com/apps/intl/en/terms/sla.html, 2020.