July 2020

Network Function Virtualization In Fog Networks

Nazli Siasi
University of South Florida

Network Function Virtualization In Fog Networks

by

Nazli Siasi

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Nasir Ghani, Ph.D.
Sylvia Thomas, Ph.D.
Zhuo Lu, Ph.D.
Mehran Mozaffari, Ph.D.
Adrian Jaesim, Ph.D.

Date of Approval:
July 3, 2020

Keywords: Virtual Network Function, Provisioning Problem, Fog Networks

Acknowledgments

I would like to express my sincere appreciation to my mentor and advisor, Professor Nasir Ghani, for his tremendous support, patience, encouragement and guidance during my doctoral studies on the personal and technical sides. His great support not only gave me the direction to march forward on the academic road but also helped me make a clear career plan. I am deeply indebted to him for giving me the opportunity to work with his group and for supporting me with teaching and research assistantships.

I would like to thank my dissertation committee members, Professor Sylvia Thomas, Professor Zhuo Lu, Professor Mehran Mozaffari Kermani, and Professor Adrian Jaesim, for their valuable feedback and guidance that improved this work. Special thanks to Professor Chris Ferekides and the department staff for their support during the period of my graduate studies at USF.

My former and current research colleagues, Dr. Adrian Jaesim, Dr. Diogo Oliveira, and Aldin Vehabovic for their encouragement and collaboration.

Dedication

To my mom Faegheh, for her endless love, support and patience while being far from her daughters,

To my dad Ali, for being my first teacher, for encouraging me to believe in myself, and teaching me to pursue my passion and dreams and overcome whatever difficulties I may face in life.

To my soulmate, my sister Shalaleh, for being my inspiration and encouraging me to go on every adventure, especially this one, and to my brother-in-law Shahin, for his continuous support.

To my grandparents Navab and Hossein, whom I promised to make proud by pursuing my doctorate degree. I hope that I have fulfilled this promise to them and very much wish that they were still alive today to share with me the celebration and the success of my studies.

Table of Contents

List of Tables

List of Figures

Abstract

The *network function virtualization* (NFV) paradigm uses commodity servers to implement "softwarized" networking capabilities and replace costly, proprietary hardware systems. In particular a wide range of *virtual network function* (VNF) tasks can be implemented here, including firewalls, load-balancers, encryption engines, address translation devices, domain name servers, even routers and switches. NFV also enables a high-degree of service flexibility, allowing operators to interconnect multiple VNFs to build highly-customized *service function chain* (SFC) sequences for their clients. As a result these related technologies are gaining widespread traction in enterprise and cloud-based settings, offering unprecedented service agility and cost effectiveness. Researchers have also investigated a wide range of VNF placement and SFC provisioning strategies to achieve various operator and client objectives.

Nevertheless, most studies on VNF placement and SFC embedding have only considered operation across larger cloud-based settings. These infrastructures are comprised of large core datacenter sites with abundant storage and computational resources. However cloud computing is not well-suited for highly time-sensitive real-time services and applications, i.e., with tight delay and delay-jitter requirements. Localized contextual data support is also quite problematic, e.g., for services such as weather or traffic. In addition, cloud-based services also increase traffic demands across the network along with vulnerability to remote failures and outages. It is here that edge/fog computing paradigms offer much promise by placing smaller storage and computing pools closer to the end-users. These designs can

provide much lower service delays, improved localized information support, and reduced network bandwidth overheads. Hence there is a growing need to extend NFV provisioning by fully leveraging fog-based infrastructures to properly support stringent end-user service needs. This remains a largely open problem area today.

Overall provisioning NFV services over fog-based networks imposes some key differences versus cloud-based operation. Most notably, fog nodes have orders magnitude less resources and capabilities in terms of storage, computation, and bandwidth interconnectivity. As a result resource constraints become a critical factor, along with service delays. To address these challenges, this research dissertation presents a detailed investigation of SFC provisioning in NFV-enabled fog computing networks. Specifically, novel SFC mapping schemes are developed for the fog domain by taking into account important client parameters, i.e., delay, bandwidth, node resources, and function dependency. Furthermore, several survivability provisions are also presented to improve the reliability of these methods, i.e., including pre-fault protection and post-fault restoration methodologies. Overall these contributions provide a solid base from which to leverage NFV technologies at the network edge.

Chapter 1: Introduction

The focus of this thesis dissertation is to investigate the application of *network function virtualization* (NFV) technologies in edge/fog computing network domains. This topic area is of key importance to various end-user service scenarios today. Hence this introductory chapter presents a high-level background and introduction to the problem space and then identifies the key motivating factors for this research work. The major contributions of the dissertation are also presented along with an outline.

1.1 Background

Network operators have traditionally used proprietary vendor-specific hardware and software systems to build their physical networks and offer services to their end-user clients. However these legacy setups exhibit a high degree of hardware-software dependency and coupling, leading to a generalized lack of service flexibility and very costly and complex upgrade cycles [1]. Given the increasing diversity of new networking services and the wide-ranging *quality of service* (QoS) requirements of many users, these legacy setups are proving to be an immense challenge today.

In light of the above, network designers and standards associations have evolved a new set of paradigms for both the network control and data planes in and effort to streamline network design and operation. Most notably, the *software-defined networking* (SDN) concept was introduced several years ago to help centralize (condense) the networking control plane into a centralized controller entity. This transition decouples the network data and control planes and greatly reduces the complexity of switching devices, i.e., as they no longer have to run

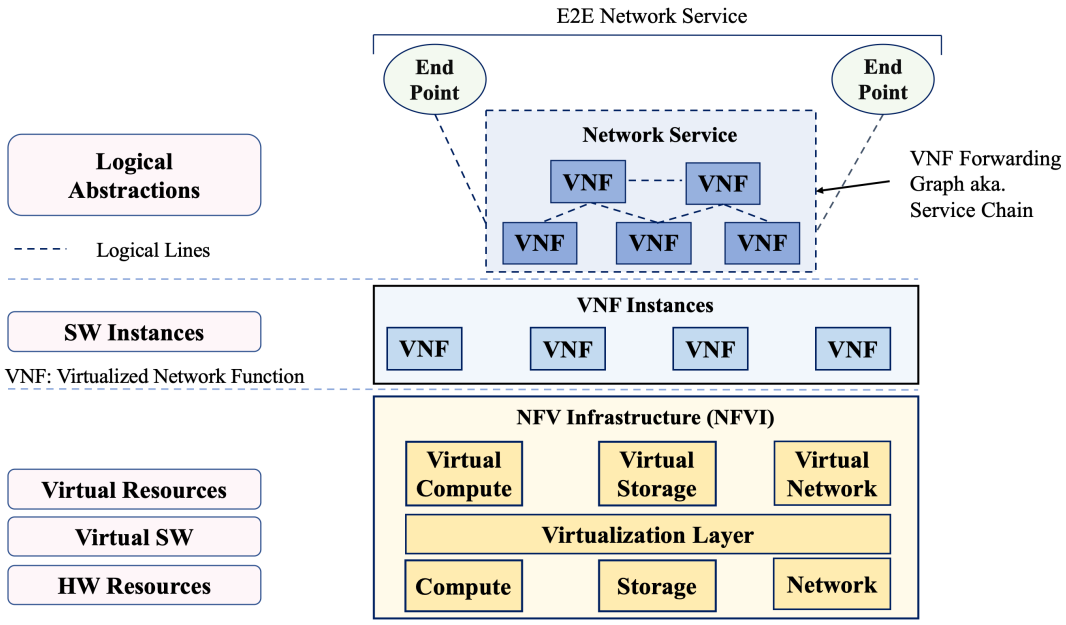

Figure 1.1: High level VNF architecture proposed by ETSI

complex distributed protocols and maintain relative state synchronization. Instead network switches can be replaced by much simpler data-plane devices which receive their forwarding rules through a northbound interface (to the SDN controller). Indeed, SDN eliminates the dependency on vendor-proprietary network control/provisioning software and offers immense potential for developing customized networking applications, e.g., for traffic engineering, survivability, security, management, billing, etc. As a result this technology is seeing wide scale traction and deployment in datacenter, enterprise, and even larger metro/wide-area networking domains.

Meanwhile NFV paradigms have also been evolved to advance the data plane and reduce dependency/cost on vendor-specialized hardware switching and routing platforms. The main idea behind this approach is to separate physical layer devices from the *network functions* (NF) that they run. Namely, software-based *virtual network functions* (VNF) can now be defined as application instances running on commodity *commercial-off-the-shelf* (COTS) servers. Some key examples here include functions such as firewalls, *network address trans-*

lation (NAT) boxes, *domain name service* (DNS) servers, load-balancers, encryption engines, even routers and switches, etc. Furthermore, multiple VNF instances can also be instantiated and interconnected to build customized function sequences, thereby provisioning a wide range of end-user client services, i.e., generally termed as *service function chains* (SFC). As such NFV greatly lowers (even eliminates) the need for vendor-proprietary networking systems. Furthermore VNF upgrades can also be done via software updates. Moreover client VNFs can also be hosted beyond the enterprise in cloud-based datacenters offering unprecedented scalability (economy of scale). Indeed such network "softwarization" is yielding immense capital expenditure (CapEx) and operational expenditure (OpEX) savings [2].

1.2 Motivations

In light of the above developments, researchers have been investigating the use of SDN and NFV paradigms to support modern network service demands. These efforts have ranged the full spectrum, including provisioning algorithm design, testbed development, and even business case analyses. In particular, most NFV provisioning studies have focused on VNF mapping or SFC (chain) embedding algorithms in generalized cloud-based networks [3]. Many of these problems are classified as NP-hard, and hence a range of heuristic and optimization schemes have been tabled.

Now the distributed cloud services model offers immense scalability and consolidation, eliminating the need for costly on-site hardware and maintenance. As a result most SFC studies have assumed abundant resource scalability at datacenter nodes. Nevertheless cloud-based provisioning is generally problematic for time-sensitive (real-time) client services and applications with tight delay and delay-jitter QoS requirements. Indeed most SFC embedding studies do not consider end-user latency requirements or localized contextual information needs. Moreover dispersed cloud-based infrastructures also increase traffic demands across the network, along with vulnerability to remote failures and outages.

In light of the above, more refined edge/fog computing paradigms offer much promise for NFV support [4],[5],[6]. Namely, fog computing is an evolution of cloud computing that places smaller storage and computing pools closer to the user edge. The intent here is to allow service providers to leverage these resources (with closer proximity to end-user clients) to reduce service latency and also support more localized contextual information needs. The latter capability is particularly relevant for mobile user services requiring access to information on weather, traffic, peer locations, etc. Furthermore fog-based designs also lower overall bandwidth usage requirements across the wider metro/core network.

However fog nodes have orders magnitude less resources than centralized cloud data-centers, i.e., in terms of storage, computation, and bandwidth interconnectivity with other nodes. Hence resource constraints become a critical factor when trying to provision services in these environments. As a result many studies have looked at resource provisioning in "hybrid" fog-cloud networks, e.g., including topics such as content placement, network congestion, operational costs, network traffic engineering, and security, see studies in [7],[8]. Despite these contributions, the further study of NFV service provisioning in fog networks, particularly SFC demands, has not been addressed in much detail. Indeed this is a relatively new topic area with strong relevance to emerging service needs.

1.3 Problem Statement

To address the above challenges and concerns, this thesis focuses on the study of SFC demand provisioning in fog computing networks. In particular there is a growing need to develop new methodologies that take into account key end-user requirements for real-time context-sensitive user services, i.e., including bandwidth throughput, end-to-end delay, and function dependency (ordering). In addition these solutions must also incorporate critical network operator concerns with regards to resource constraints and resource utilization. Furthermore related solutions must provide rapid provisioning of incoming requests in a

scalable on-line manner in order to ensure practical real-world applicability. Finally, there is critical need to improve the survivability of SFC demands across fog infrastructures, i.e., in order to meet the service reliability needs of high-end users. Specifically a range of strategies need to be investigated here for various fault scenarios, i.e., including pre-fault protection strategies for single node/link faults and post-fault restoration methods.

1.4 Proposed Work and Contributions

In light of the above, this dissertation addresses the important emerging area of SFC provisioning and survivability in fog networks. The main contributions of this work include the following:

- Novel polynomial-time on-line heuristic algorithms for embedding SFC demands into multi-layer fog infrastructures (including methods based upon delay and load minimization).

- New pre-fault SFC demand protection heuristics to improve service survivability in fog-based infrastructures, i.e., by handling the most prevalent case of single node or link failures.

- Novel post-fault SFC restoration schemes to further improve the resource efficiency of SFC demand survivability in fog networks (including methods based upon end-to-end and intermediate recovery).

The rest of this thesis is organized as follows. First, Chapter 2 presents a detailed survey of SFC provisioning for NFV with a focus on fog networks. Some NFV survivability methods are also reviewed here. Next, Chapter 3 presents a novel heuristic framework for SFC embedding in realistic two-layer fog infrastructures. Chapter 4 then addresses the critical topic of SFC survivability in fog networks and augments the above solution to support pre-fault protection

of SFC demands (to overcome single node and link failures). Extending upon this, Chapter 5 considers complimentary post-fault SFC restoration strategies to achieve broader resiliency. Finally, Chapter 6 summarizes the key research findings of this dissertation and presents overall conclusions and promising areas for future study. Additional appendices also provide a glossary of acronyms and listings of all key variables used in this thesis.

Chapter 2: Literature Survey

Cloud computing relies on large datacenters to support a diverse range of computational needs for end-user applications. Nevertheless despite its immense scalability, this paradigm is not well-suited for delay-sensitive services owing to the increased latencies of remote cloud computing sites. In response network designers have evolved more specialized fog computing paradigms to support real-time and localized service needs. At the same time emerging technologies such as NFV are also leveraging the cloud to redefine modern data network designs. Namely, atomic networking device functionalities are now being implemented in software, helping streamline service provisioning and limit the need for specialized hardware systems. Hence there is significant potential to deploy NFV-based capabilities in the fog computing domain.

In light of the above, this chapter presents a detailed look at these key technology spaces. Foremost, the NFV architecture is reviewed, along with a survey of research studies on VNF placement and SFC mapping. Subsequently the fog computing concept is detailed along with a review of recent work on NFV provisioning within such domains, i.e., including survivability schemes. Some key open problem areas are then identified (based upon these surveys) in order to properly motivate this research effort.

2.1 NFV Architectural Overview

As noted in Section 1.1, the main idea behind NFV is to extract atomic networking functions and implement them in software on commodity servers (typically located in enterprise or cloud datacenter environments). Termed as VNFs, these entities can either be run

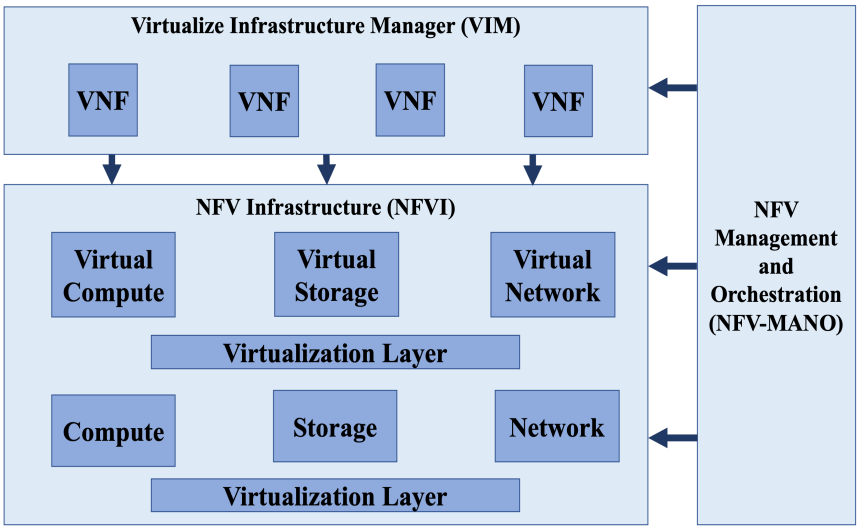

Figure 2.1: High level VNFM architecture proposed by ETSI

in virtualized hypervisor or dockers environments to implement key functionalities such as firewalling, DNS lookups, caching, load balancing, QoS support, etc. Overall, NFV delivers tremendous service flexibility and scalability and allows operators to rapidly deploy new services comprised of customized VNF sequences, termed as SFCs. Moreover VNF upgrades can also be done by using simple software updates as opposed to slow and costly hardware upgrades [2].

The standardization of NFV is being led by the European Telecommunications Standards Institute (ETSI), which published its first architectural specification in 2013. This standard introduced the notion of a *NFV infrastructure* (NFVI) which includes all the hardware and software components used for hosting VNF entities. Specifically the NFVI is segmented into three layers, as shown in Figure 2.1 from [9]. In particular these are:

- Physical Layer (Layer I): This layer implements packet transmission and reception and consists of hardware platforms interconnected by physical links, i.e., COTS servers, networking switches and routers, and transmission links. These entities can be located in larger core datacenters and/or edge and enterprise sites

- Intermediate Virtualization Layer (Layer II): This layer acts as bridge between the physical and logical layers by mapping VNFs to underlying physical platforms. Namely, Layer II consolidates VNF applications (Layer III) onto server machines running on hardware server platforms (Layer I). This layer also leverages virtualization techniques to decouple NF software from the hardware systems supporting computation, storage, and networking functionalities (as opposed to dedicating devices to VNFs).

- Logical Layer (Layer III): This layer consists of various service functions and is controlled by a *VNF manager* (VNFM). The logical layer processes *service function requests* (SFRs) by assigning them to one or more service functions with various tasks and capacities. Service chaining is also supported by combining several coexisting VNFs to build a service. Namely chaining defines connectivity and execution sequences (dependency) between service functions via forwarding graphs. Note that VNF instances do not need to be aware of virtualization environments or physical layer devices.

Overall NFV can implement a wide range of control and data plane functions across multiple networking technology domains, e.g., such as enterprise, datacenter, cellular, wireless sensor, IoT, vehicular, etc. For example, some key enterprise-level functions can include virtual firewalling, virtual routing and switching, address translation, load balancing, monitoring, billing, etc. Meanwhile, NFV-based solutions are also being considered for emerging 5G network architectures which define a functional decomposition between *remote radio head* (RRH) and *baseband unit* (BBU) systems. Specifically, BBU functionalities can be implemented via VNFs running at datacenter sites to support functions such as mobility management, session and policy management, *serving gateways* (SGW), and *packet data network gateways* (PGW). To date a wide range of technical studies have looked at NFV provisioning concerns, and some of these are now detailed.

2.2 Survey of NFV Resource Provisioning

One of the main considerations in NFV provisioning is the placement of VNFs onto physical infrastructures. Accordingly, researchers have investigated the problem of mapping, or embedding, network functions over underlying datacenters where the key resources include server memory, processing compute power, and link bandwidth. Now the VNF embedding problem is NP-hard [10] and is of similar complexity to the *virtual network embedding* (VNE) problem. Meanwhile a related but more challenging NFV problem is the embedding of service chains, i.e., SFC provisioning problem. Namely a SFC consists of a sequenced interconnection of multiple VNFs, typically mapped onto different datacenters, to deliver an end-to-end client service. In general SFC provisioning is also classified as NP-hard [11] but is more complex than VNE, i.e., since it requires both setup of a virtual infrastructure (NFVI) and embedding of VNFs across nodes. In response researchers have proposed a range of SFC provisioning solutions. Consider some further details here.

Overall SFC provisioning is a well-studied problem and involves two key steps, i.e., VNF mapping and connection routing. Accordingly, most schemes can be classified as either static or dynamic. Namely the former assume that the full set of incoming demands is known in advance (a-priori) and proceed to map them in an off-line manner using optimization-based methods. Meanwhile the latter assume that demands arrive in a random "on-demand" manner and use faster heuristics type methods. Although dynamic strategies are better suited for real-world settings, these methods pose added challenges in terms of resource efficiency, i.e., since future demands are not known. Overall most SFC provisioning algorithms are designed to achieve a particular provisioning objective. For example this can include minimizing certain quantities (such as resource utilization, deployment cost) or maximizing other quantities (such as available resources, carried load, reliability, energy efficiency). Various studies are now reviewed, and interested readers are also referred to surveys in [12],[13],[14],[15].

The work in [16] proposes a resource allocation architecture for jointly solving the VNF mapping and connection routing sub-problems. Here the authors implement initial resource allocation followed by global resource re-allocation with the aim of minimizing energy consumption (while considering delay and server utilization). VNF placement is then formulated as an *integer linear programming* (ILP) problem for smaller network sizes, and a series of on-line and off-line (near-optimal) heuristics are also proposed for handling larger scenarios. Overall findings show fast execution times with the latter schemes, i.e., sub-second ranges for the scenarios tested. Meanwhile [17] investigates SFC provisioning and tables an ILP optimization model for VNF placement across geographically-distributed clouds. This work also tries to minimize response times and incorporates other constraints such as deployment cost and energy consumption. Also, [18] looks at joint VNF placement and routing of services with traffic engineering constraints. Namely a detailed ILP model is presented along with a greedy heuristic to maximize the number of satisfied VNFs, i.e., while minimizing VNF setup and routing costs. Load balancing is also done by incorporating link capacity constraints. Results show that both methods increase the number of satisfied VNFs without overloading links.

Many studies have also looked at dynamic on-line SFC placement strategies. For example, [19] proposes a *mixed ILP* (MILP) formulation to perform VNF placement and path routing of a single incoming demand. The objectives here are to reduce network delays and lower resource utilization at nodes and links. Although the computational complexity of the optimization model grows exponentially with the number of flows, this scheme can still handle a notable number of demands since it performs incremental (on-line) provisioning. Results also confirm adequate reduction in network link and node utilization. Meanwhile [20] presents an on-line solution that tries to maximize the number of satisfied requests using a competitive ratio. Namely this ratio is defined as the worst-case ratio between the cost of the solution found by the algorithm and that of the optimal solution. Results indicate

that the proposed on-line algorithm is asymptotically optimal in case of deterministic and randomized situations (lower bounds for an off-line approximation are also studied, and an ILP formulation is presented to show that the problem is NP-complete). Additionally, [21] proposes an on-line solution for SFC mapping and scheduling in which VNFs can share a single VM. Several greedy algorithms are proposed based upon various criteria, e.g., VNF processing times, available buffer sizes at nodes, etc. A competing local *Tabu search* (TS) algorithm is also used to iteratively improve upon a randomized initial solution. Findings show that the latter scheme gives slightly better results. Nevertheless this study does not consider link delays and bandwidth availability on links. In other work, [22] presents an on-line heuristic for dynamic provisioning and placement of multiple concurrent VNFs in datacenters in smaller intra-campus settings. The objective here is to find a tradeoff between the requested number of instances of each VNF to provision and their deployment and operational costs. Accordingly two solutions are proposed, i.e., for single and multiple SFCs. In the former a randomized algorithm is presented to achieve a competitive ratio by taking into account available server resources and current traffic rates. Meanwhile a minimal weight matching algorithm is used in the latter scheme to minimize deployment cost by avoiding frequent VNF deployment at datacenters.

Various studies have also looked at network latency and delay in SFC provisioning for time-sensitive services. For example [23] uses real-world measurements to incorporate network latency for VNF placement. The goal here is to minimize delay and required lead times for allocating VNFs to a VM (versus simply mapping to hosts based upon available resources). Results confirm reduced delays (versus some best-fit and first-fit heuristics) as well as reduced VNF lead times. The authors in [24] also detail a dynamic VNF placement heuristic that accounts for network capacity and tries to reduce response times for SFC requests. However this work only considers transmission delays, which tend to dominate in larger core clouds or multi-cloud settings, i.e., and not other queuing and processing delays.

Another delay-aware heuristic is also outlined in [25] for VNF placement in large infrastructures with static demands. The goal here is to minimize the number of mapped VNF instances in order to ensure that end-to-end SFC latency constraints are met. Note that end-to-end delay is considered as the sum of all path delays and processing times, including link transmission delays and VNF processing delays. Finally, [26] studies VNF placement for different applications with a focus on end-to-end delay guarantees. Namely, a *mixed-integer quadratically constrained program* (MIQCP) formulation is developed to model the linear dependency between the amount of resources allocated to a VNF and its processing delay. However this work lacks scalability as it only treats a small number of demands.

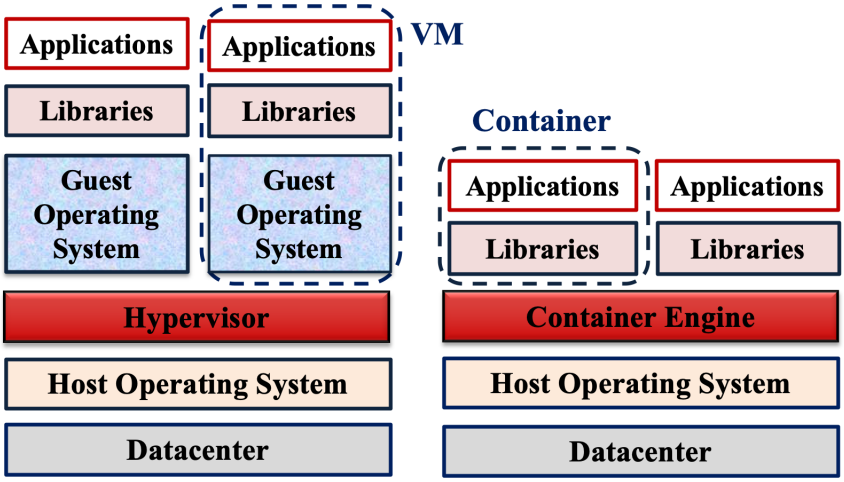

Figure 2.2: Hypervisor versus container-based virtualization

Other studies have also looked at dynamic SFC provisioning. For example [27] outlines a VNF dependency-based SFC routing and mapping scheme. Namely container-based environments are leveraged to improve utilization and lower cost, Figure 2.2 from [27]. The scheme performs batch SFC mapping of demands with functional dependencies between VNFs and uses shortest path routing to reduce SFC delays. Overall results show improved performance over VM hypervisor-based mappings. Additionally [28] also demonstrates SFC orchestration in SDN-controlled environments, i.e., including dynamic VNF selection and intent-based

traffic steering. Namely SFC requests (with differing bandwidth and latency requirements) are send to a chain optimizer module which runs a CPLEX solver to compute mappings to minimize end-to-end latency. These results are then sent for instantiation over a set of cloud domains. Finally researchers have also studied "dynamic" SFC chains where VNF sequences can vary with time, e.g., to support potential scenarios such as video streaming, autonomous vehicle applications, etc [29]. Clearly these types of SFC demands pose even more challenges for service providers trying to optimize network resource allocation.

Overall most of the above NFV studies assume that VNFs will be hosted in cloud-based server environments. As noted earlier, the distributed cloud model offers immense scalability and consolidation, eliminating the need for costly on-site hardware and maintenance. Accordingly nearly all SFC schemes assume abundant resource capacities at underlying datacenter nodes. Despite these saliencies, cloud-based NFV architectures are problematic for real-time end-user services and applications with tight delay and delay-jitter requirements. In general most SFC studies do not consider end-user latency requirements or localized contextual information needs when performing SFC embedding. Moreover highly-dispersed cloud infrastructures will also increase the likelihood of service vulnerability to distant failure and outage events. It is here that modified fog computing paradigms [30] offer much promise, as discussed next.

2.3 Overview of Fog Computing

Edge computing is a distributed networking paradigm which hosts smaller computation and storage pools at multiple dispersed edge locations [5]. By and large this concept is a non-trivial extension of the cloud, and hence researchers have proposed further variants such as fog computing [6] and virtual cloudlets [31]. In particular fog computing was originally proposed by Cisco [4] to address the shortcomings of cloud-based architectures, mainly delay-related QoS requirements [32]. This paradigm is ideal for handling a larger number of edge

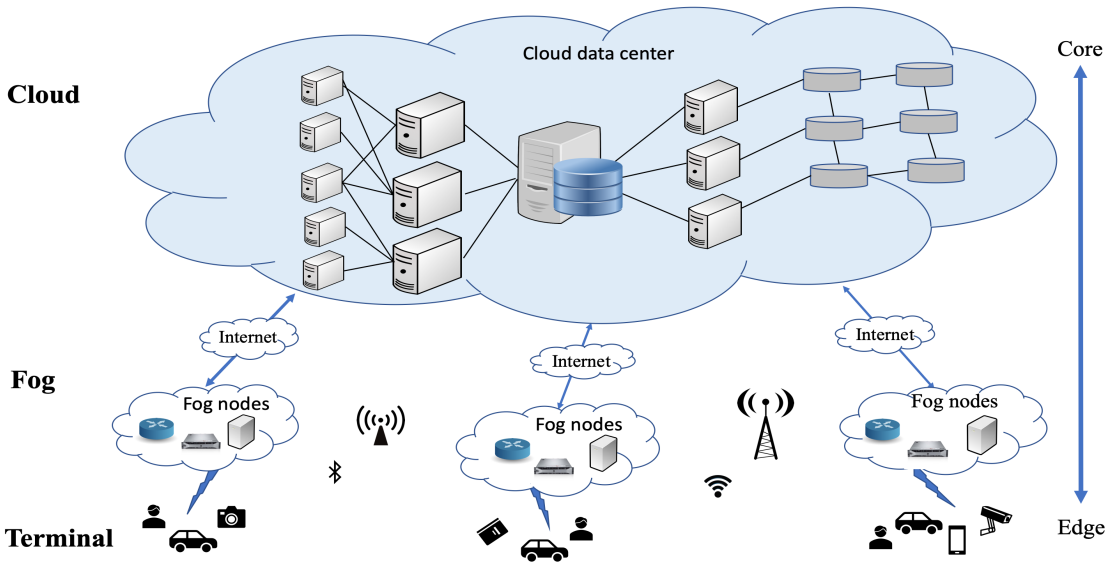

Figure 2.3: Conventional hierarchical three-layer fog computing architecture

devices with limited energy resources, e.g., such as sensors and smartphones. Fog computing is also very suitable for more localized application support as well as IoT application scenarios with a large number of sensor nodes.

To date, several reference architectures have been proposed for fog computing. Most of these designs assume a three-layer hierarchical setup, termed as the conventional architecture, as shown in Figure 2.3 [33]. In particular, this architecture consists of the following layers:

- Terminal layer: This layer is the closest to the end-users and consists of terminal devices such as mobile phones, sensors, smart vehicles, and specialized IoT devices. In general these devices are geographically distributed and responsible for sensing data (about physical objects or events) and transmitting this information to upper layers for further processing and storage [34]. Terminal layer devices can also be connected to one or more fog nodes through various types of wireless access networks, e.g., such as cellular, wireless local area, wireless personal networks (ZigBee), etc.

- Fog computing layer: This layer serves as an intermediary layer between terminals and large cloud datacenters and consists of multiple fog nodes. These nodes have

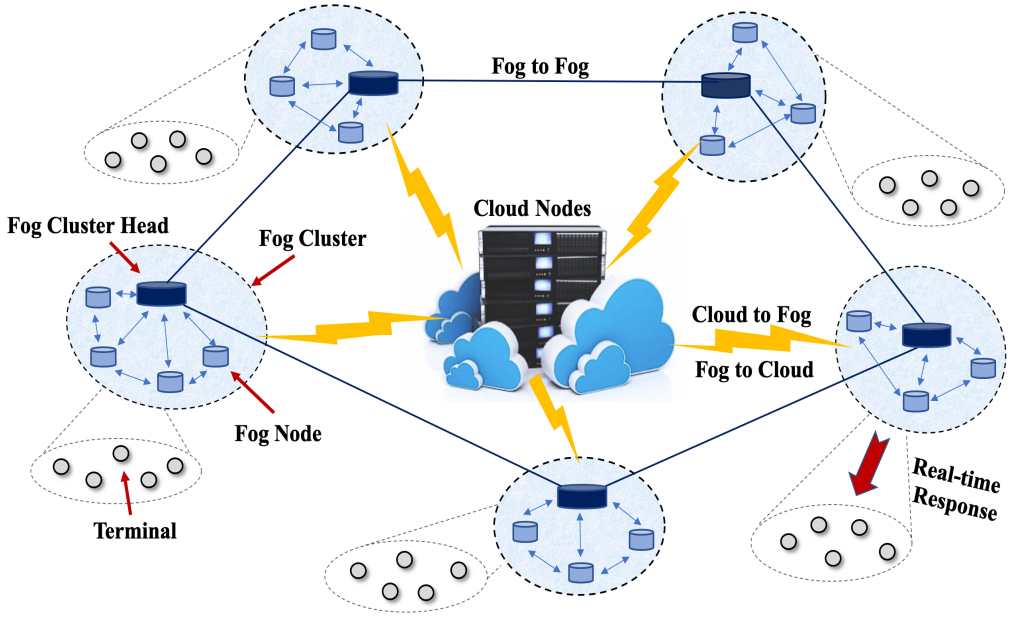

Figure 2.4: Hybrid fog-cloud architecture with fog nodes and fog cluster heads

much smaller resources pools than cloud (or enterprise) datacenters and are also interconnected through intermediate networking devices, e.g., such as routers, gateways, switches, access points, base stations, etc. Hence terminal devices communicate and connect with fog nodes to obtain services (such as computing, storage, and data transmission). Fog nodes also use IP routing infrastructures to interconnect/integrate with larger cloud datacenters offering more scalable services support. Now in the generalized case, fog nodes can also be static or mobile [35].

- Cloud layer: This layer is composed of large datacenter sites hosting massive numbers of high-end storage and compute servers. These sites can be further interconnected via high-bandwidth routing networks and network virtualization software to provide seamless fabrics. Overall the cloud layer provides extensive computational scalability and long-term data storage capabilities. As a result many different end-user services and applications have already migrated to the cloud in recent years.

Leveraging from the above, others have also proposed some modified four-layer fog-cloud architectures by adding more selectivity at the fog layer. For example, Figure 2.4 from [36] shows a sample architecture with two fog-based layers. Namely regular fog nodes (on the lower layer) interface with terminals in Layer I and are designed to support VNF demands with lower processing requirements. Meanwhile larger fog cluster heads (on the upper layer) support higher capacity VNF processing needs and also interface with cloud datacenters.

Further alternative fog models have also been proposed. For example, [37] outlines a detailed "top-to-bottom" layered design comprising of six layers, Figure 2.5. At the lowest level, the physical and virtualization layer is composed of physical terminals and virtual nodes. Above this a monitoring layer is defined to handle requested tasks/functions and monitor energy consumption. Meanwhile the pre-processing layer manages data management related tasks (such as analysis, filtering, reconstruction, and trimming). Overlying the pre-processing layer is a temporary storage layer which hosts data for limited time intervals for the purposes of distribution, replication, and re-duplication. A security layer is also defined to perform critical functions such as data encryption/decryption, privacy, and integrity checking. Finally, the highest transport layer is responsible for sending data to the cloud. Another specialized fog computing architecture is also outlined in [38], as per Figure 2.6. This framework is designed for smart cities (with a large numbers of infrastructure components and services) and specifies four layers, i.e., physical sensors, upper and lower fog layers, and cloud. Specifically lower fog layer nodes receive raw data from the physical sensors, whereas upper fog layer nodes provide intermediate computation. Overall this four layer fog architecture can provide rapid response times at the neighborhood, community, and city-wide levels.

In general fog computing offers a wide range of benefits including:

- Reduced service delays: Processing user services locally at edge nodes, instead of at core datacenters, can greatly lower round-trip times for time-sensitive applications.

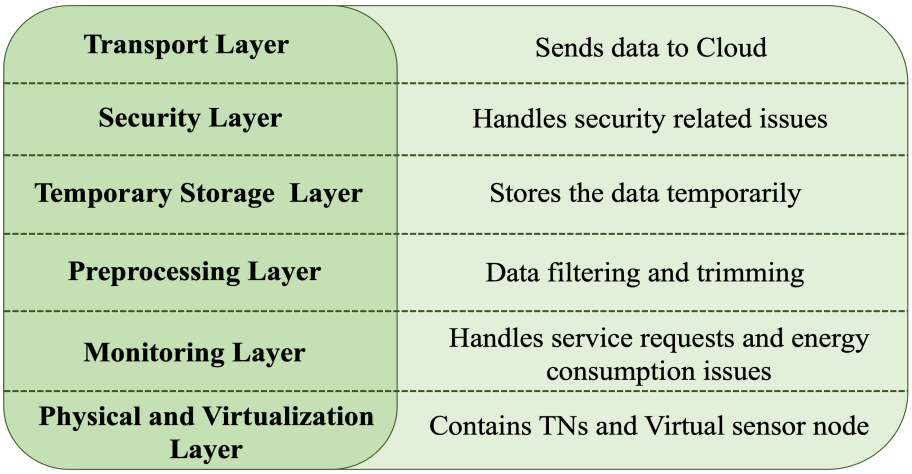

Figure 2.5: Layered fog computing architecture

Some sample edge processing tasks can include computation, data storage, request redundancy removal, etc. For example high-demand content can be pre-fetched and cached at fog nodes.

- Improved bandwidth efficiency: Since fewer computations are relayed to the cloud, fog computing also lowers the amount of traffic (and resource utilization) across larger metro and wide-area core networks. This reduction is a key advantage as traffic volumes continue to scale over time.

- Lower energy/power consumption: Reduced data transfer across large network domains also yields sizeable energy savings. This reduction enhances the migration to green networks and notably lowers operational costs. Accordingly some studies indicate 40% lower energy consumption versus traditional cloud setups [39].

- Increased reliability: Service reliability can be modeled as inversely proportional to the number of systems utilized. Given the closer proximity of fog nodes to end-users, associated traffic flows will now traverse much fewer links, switches/routers, and processing nodes. Hence fog-based service reliability will generally be higher.

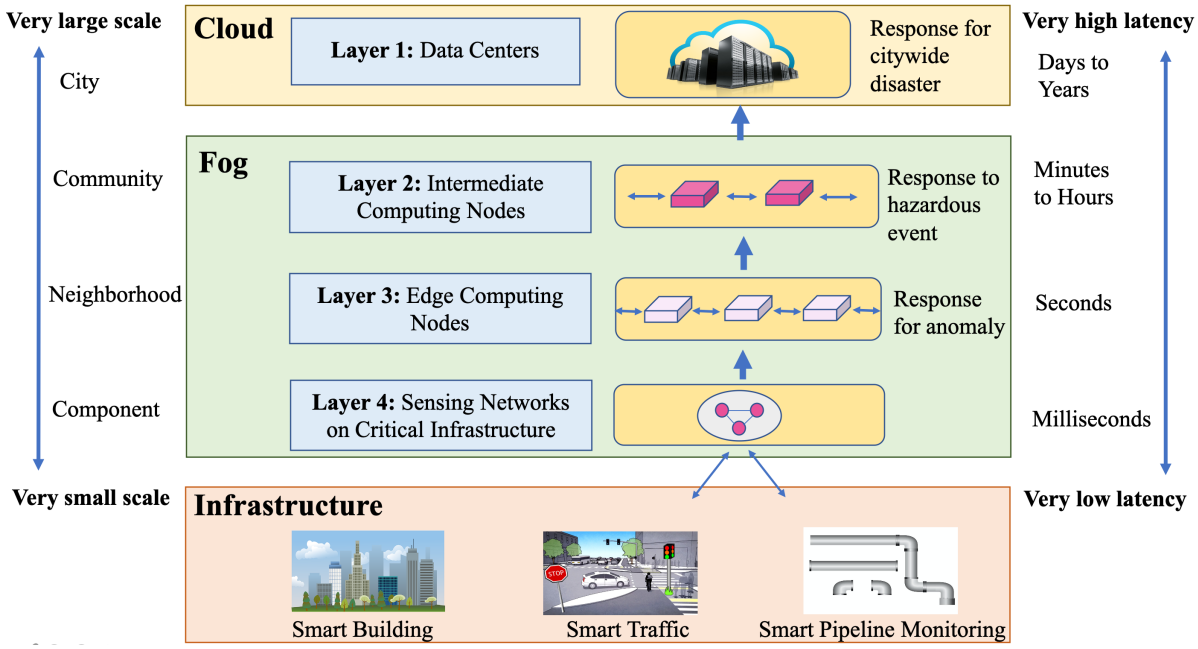

Figure 2.6: The 4-layer fog computing architecture for smart cities

- Improved coverage and mobility support: Proper deployment of fog nodes at strategic locations can help improve service quality for mobile end-users, e.g., along highways and roadways, on cellular base stations, on museum floors, in subway stations or other points of interest, etc. In particular user response times can be greatly reduced for highly-mobile terminals on moving platforms. Furthermore fog node themselves can also be mobile or static.

- Location awareness: Location tracking (either active or passive) can be used to provide a high degree of localized contextual information. Indeed this data is vital for certain end-user services and applications, e.g., such as weather or vehicular traffic mapping. Many IoT applications are also dependent on localized data [34].

- Traffic aggregation: In certain use case scenarios, correlated sets of data from a group of end-users can be processed by a reduced number of fog nodes. Similarly cloud datacenters can also forward highly-correlated traffic to specific fog nodes for localized

processing and distribution. This aggregation helps reduce the volume and routes of data flow across the network [40],[41].

- IoT applications support: IoT deployments continue to grow across a wide range of sectors. As are result distributed location-aware fog architectures are being used to manage large numbers of sensors generating constant data streams (which are difficult to transfer to centralized cloud sites). Some typical application scenarios here include environmental monitoring, power grid management, water treatment, and automatic traffic management, etc [42].

- Improved security: Fog nodes can also enhance the overall security posture of service providers. Namely multiple dispersed nodes provide more sites for implementing critical distributed security functions, e.g., such as access control, encryption, intrusion detection/protection, etc. Fog setups can also lower the risks associated with system-wide upgrades as compared to traditional setups using complex/slower *over-the-air* (OTA) firmware updates. Instead one only needs to update the algorithms or micro-applications running on fog servers [35].

Despite the above saliencies, fog nodes generally have (orders magnitude) less storage, computing, and bandwidth resources as compared to cloud datacenters. Hence resource constraints become a critical factor when provisioning services in these environments. As a result network designers have also proposed hybrid fog-cloud architectures to leverage the benefits of both paradigms. In particular a range of research problems have been studied here, including content placement, network congestion and traffic engineering, and security, see studies in [7],[8].

However the increasing prevalence of network softwarization and virtualization paradigms is facilitating new capabilities in the fog. Specifically SFC enables rapid provisioning of customized delay- and context-sensitive services at much lower price points. Nevertheless

the re-application of existing cloud-based SFC provisioning algorithms (Section 2.2) in the fog domain is problematic for several key reasons. Foremost, the assumption of abundant resource pools does not hold for constrained fog nodes. The lack of latency parameters in most SFC provisioning algorithms also limits their ability to leverage the tiered nature of fog-based setups for delay-sensitive users. As a result recent studies have started to investigate more specialized SFC provisioning algorithms for the fog domain. This is a relatively new area and a few related works are surveyed here, see also Table 2.1.

2.4 Survey of NFV Provisioning in Fog Computing

Researchers are starting to introduce NFV-based concepts in the fog domain. For example, an earlier study in [43] proposes a *platform as-a-service* (PaaS) architecture to support applications in hybrid cloud-fog settings. However this effort only considers a single VNF type and does not implement VNF placement in the fog layer. Meanwhile the authors in [44] study healthcare IoT application support across components spanning the cloud and fog domains. Nevertheless this work only focuses on mechanisms for control, signaling, and data interfaces between the cloud and fog domains, i.e., and not detailed provisioning algorithms. A more detailed study on resource provisioning for IoT applications in the fog is presented in [45]. Here the authors develop an optimization model that takes into account applications response times, application deadline violations, and resource heterogeneity. This work also applies the concept of fog colonies with micro-datacenters composed of an arbitrary number of fog nodes, i.e., to decentralize processing network overheads and multi-cloud deployment costs. The formulation is solved using greedy first-fit heuristic and genetic algorithms to show reduced communication delays and improved resource utilization. Also, [46] details two greedy heuristics to efficiently map application services to fog nodes and minimize cost. The first scheme prioritizes the deployment of high-demand services on fog nodes and releases

low-demand services, whereas the second scheme focuses on increasing revenue. However this work only considers a single arriving request and lacks provisions for SFC demands.

Furthermore the work in [47] studies application component placement in NFV-enabled hybrid fog-cloud setups. Here VNF placement is done using an ILP model, and the scheme is evaluated for smaller networks to achieve cost minimization. However this work only considers a single VNF type. By contrast [48] presents a multi-layer fog and cloud architecture for video streaming applications. This study analyzes three fog layers, i.e., classified based upon their coverage, computational and storage capacity. However this work does not address SFC demand provisioning. Additional work in [49] presents an optimization model for fog-based VNF placement for IoT applications. This solution considers heterogeneous QoS requirements, and a heuristic scheme is also proposed to lower delay and improve utilization. However this work does not account for node delays for processing and queuing. The work in [50] also proposes a delay-aware SFC provisioning solution for multi-layer fog networks. Namely greedy heuristics are developed to implement various operator or client provisioning objectives, e.g., such as delay, cost, or load minimization. Findings show that load minimization gives the highest carried load. Similarly, [51] presents another solution for SFC provisioning in fog networks. Here a modified Tabu search is proposed for VNF mapping along with shortest path routing for service chain connections (taking into account delay and load balancing efficiency). Results show reduced delays and energy consumption versus cloud-based SFC mappings. Finally [52] tables a combined SDN and fog architecture to provide more scalable and flexible solutions for *vehicular adhoc network* (VANET) with delay-sensitive and location-awareness services.

Researchers have also proposed the integration of NFV (and SDN) concepts with fog infrastructures for next-generation 5G wireless networks. For example [53],[54] address service chaining in hybrid fog-cloud setups and focus on higher-level architectural issues relating to virtualization and security. However these frameworks assume abundant resources at

the (single) fog tier and do not address delay-sensitive applications needs. Other work in [55] a user-centric scheme to split VNF demands between BBU and RRH nodes nodes in emerging 5G cloud-based *radio access network* (RAN) environments. Namely this scheme leverages fronthaul capabilities and computational resources to ensure efficient baseband function placement for each user in terms of throughput. Baseband processing chains are also treated as composite VNF entities. Hence by partially centralizing the BBU, bandwidth and latency requirements can be relaxed and fronthaul costs reduced. Meanwhile [56] also integrates NFV with *fog access points* (F-AP) to improve handovers in 5G networks. Namely this solution leverages edge caching and virtualization at F-AP nodes to reduce handover signaling costs/overheads. Similarly the work in [57] develops a centralized BBU pool for VNF placement to accommodate stringent delay constraints. Namely a graph-based clustering approach is used to split/place baseband VNFs, along with a genetic algorithm scheme to reduce fronthaul costs.

2.5 Survivability Design for NFV

Most networks operators have relied upon a range of "overbuild" protection strategies to provide very high-grade "telco-level" guarantees for their legacy services, e.g., such as leased lines. However as more operators start to deploy NFV-based infrastructures, these stringent requirements are also starting to emerge for SFC-based services. As a result there is now a growing body of research on failure recovery in NFV infrastructures, and some of these contributions are surveyed.

The authors in [58] discuss generalized NFV architectures for failure recovery to improve end-to-end service resiliency. The focus here is on single node or link failures, and the work discusses the merits of resilience at the physical or virtual layers, i.e., in terms of network setup costs and failure coverage. A more detailed reliability-aware routing optimization framework for NFV is also proposed in [59]. Specifically a MILP formulation is

used to implement VNF placement and limit end-to-end delays and maximize the reliability of supported NFs. Node failures are also handled by provisioning backup VNFs on alternate paths. However findings show a direct tradeoff between reliability and bandwidth usage (and computational complexity). Meanwhile [60] presents an availability-aware SFC placement scheme using a redundancy sharing mechanism that leverages VNF multi-tenancy to increase network utilization. However this approach suffers from increased resource utilization, cost, and energy consumption. A proactive strategy for node failure recovery is also proposed in [61] by using backup paths with distinct priorities, thereby precluding post-fault coordination. Once a failure occurs, recovery is done locally by the switch attached to the failed link with minimal delay. As expected this strategy lacks SFC-level selectivity and mandates link-level redundancy, leading to increased cost and reduced network capacity.

The authors in [62] propose a shared path protection scheme for single link or node failures. Sharing is also used to reduce the amount of backup resources required (versus dedicated protection). Furthermore only storage units are replicated at the backup nodes, whereas processing units are instantiated only after a failure. However this scheme suffers from high duplication overheads for larger networks and lacks delay and cost models. Meanwhile [63] proposes a SFC routing and VNF deployment scheme for simultaneous and multiple VNF failures. This solution tries to minimize computing and resource costs by using k-node disjoint shortest paths, along with set-cover techniques for mapping. Nevertheless, the scheme is only analyzed for a limited number of VNF failures and does not incorporate SFC delay bounds. The work in [64] also proposes a shared protection reliability enhancement scheme to reduce the cost of redundant VNFs. The reliability problem is formulated as a MILP optimization to compute the optimal reliability that can be achieved at minimum cost. Reliability-aware and minimum-cost genetic algorithms are proposed to address computational complexity concerns with the MILP model. However this work only treats node failures (not link failures) and lacks restoration mechanisms. A decision tree approach for

VNF recovery from link failures is also proposed in [65]. Namely a Monte-Carlo tree search is used to select reliable links in a preventive manner and re-map the failed links in a reactive manner. However this work does not consider node failures and only looks at VNF recovery, i.e., not SFC dependency. Meanwhile a reliability-aware routing and resource sharing scheme for NFV service chains is proposed in [66]. This solution uses an ILP model to incorporate reliability constraints and also presents a more scalable (sub-optimal) greedy shortest path-based heuristic scheme. The work in [67] focuses on backup resource allocation for resiliency from node and link failures. Namely a heuristic algorithm is presented for survivable VNF chain placement, where redundant communication paths (between VNF instances) are allocated for fallback in case of a failure along the primary path. VNF migration is also done in case of node failures, i.e., onto nodes with sufficient processing capacities.

The notion of a *shared risk link group* (SLRG) was introduced earlier to define a-priori fault regions in large backbone networks. Namely an SRLG represents a set of network entities (nodes, links) with similar risk vulnerabilities. Leveraging this concept [68] proposes a path-based protection scheme to implement globalized re-routing of all requests after a failure event. The objective here is to minimize overall bandwidth usage. However results show overly extended routes with much longer delays and costs. In a similar theme [69] studies survivable NFV provisioning for large-scale disaster scenarios. First, a probabilistic SRLG model is used to define random multi-failure risk regions. Resilient NFV provisioning is then done using an ILP optimization approach to achieve a weighted tradeoff between various objectives, i.e., including resource usage minimization, routing cost minimization, and failure risk minimization. A sub-optimal greedy heuristic is also proposed for handling large-scale networks. Extending upon this, [70] proposes a genetic algorithm (meta-heuristic) for risk-aware VNF mapping and traffic routing that tries to improve the reliability of user services and/or reduce deployment and routing costs. However this work focuses on preventative

methods and does not provision any backup protection resources (or incorporate end-to-end SFC delay bounds).

Finally the authors in [71] use diversity coding techniques to achieve link and node failure recovery in NFV-based 5G and beyond networks. In contrast to conventional link recovery methods, this approach offers near-instantaneous recovery without added re-transmission overheads. Furthermore the scheme also provides significant reduction in routing and capacity costs, i.e., reduced number of redundant links and increased carried demand. Similarly the work in [72] also combines diversity and network coding techniques in NFV networks to implement rapid self-recovery without any connection re-routing. A self-recovery scheme for service function paths is also proposed in [73]. Specifically this approach recovers failed paths by using data plane signaling to temporarily shift the responsibility of affected service function (VNF) instances to another locations. Special service function forwarding nodes are also designated here to manage various cases of failed instances. However this solution can yield prolonged signaling and response times.

2.6 Survivability Considerations for Fog Computing

A handful of studies have also investigated survivability in fog computing infrastructures. For example [74] studies failure recovery of SFC demands in SDN-controlled fog computing networks. An ILP optimization failure re-routing model is proposed, taking into account reliability, utilization, and energy consumption constraints. A near-optimal polynomial heuristic is also designed to resolve run-time complexity concerns. Similarly the work in [75] presents another fault prevention and recovery scheme for SFC traffic in SDN-based fog networks. This efforts starts by introducing a multi-tier fog architecture comprised of fog nodes and SDN-controlled switches. An optimization formulation is then developed to jointly improve reliability and reduce the costs of flow re-routing and energy consumption. A fast heuristic is then proposed to improve the reliability of selected paths, while minimizing network conges-

tion. However this study does not incorporate some key SFC QoS requirements, particularly delay bounds. Similarly [76] presents another heuristic scheme for failure-aware SFC support in SDN-based networks. The work considers computational complexity, average probability of path failure, and link and server utilization levels. Meanwhile a multi-layer proactive fail-over management architecture for NFV services is also introduced in [77]. Foremost, a NFVI software layer is defined with master and backup VNF instances. The former handle specific NFV services, whereas the latter store real-time standby images of corresponding master VNFs. This layer also maintains current state between these two instances when traffic is flowing. Additionally a NFVI middleware layer is introduced for managing hardware parameters and VNF workloads. Finally a proactive fail-over management layer is specified for failure prediction. Specifically this layer uses vector machine learning and random forest algorithms to predict failures and implement proactive fail-overs.

Others have also looked at broader methodologies for improving overall resiliency in fog computing domains, i.e., albeit not specifically for NFV-based services. For example [78] presents a tree-based fault-tolerant approach for fog nodes using replication and non-replication techniques. The former method uses multiple fog nodes to serve the same request, whereas the latter uses different fog nodes to replace faulty nodes. For example data from a failing node is sent to its parent node in the non-replication case. However the proposed scheme is limited to node failures, and the tree-based mechanism can give increased load at parents nodes (higher network congestion and delays). Additionally the work in [79] develops a four-step protocol to manage fog node failures. This solution uses a check-pointing method to store service state information. This data is then analyzed to see if a potential failure can occur and appropriate preventative decisions are taken. In case of failure, the protocol also notifies all dependent entities to perform reconfiguration and recovery actions. Similarly [80] introduces various fault-tolerant scheduling methods for hybrid fog-cloud networks. These

schemes build upon existing cloud-based designs by mapping time-sensitive services to the fog layer using a variety of techniques (such as replication, check-pointing, and re-submission).

Additionally work in [81] proposes proactive and reactive strategies for failure recovery of networking elements in hybrid fog-cloud infrastructures. Namely a multi-dimensional knapsack problem model is used to determine the impact on key performance parameters such as service allocation times, recovery delays, and computing resources loads. Now the proactive strategy pre-allocates recovery resources for each primary resource allocation, whereas the reactive strategy only allocates resources after a failure occurs. Recovery resources can also be selected/identified prior to a failure in the latter approach. Nevertheless this solution does not consider link failures and limits protection resources to the same layer for each node, i.e., horizontally. Another failure recovery strategy is also presented in [82] for overloaded or broken mobile edge computing nodes based upon workload offloading to neighboring resources. However the availability of neighboring resources is not guaranteed by the protection strategy, and this solution only considers a small number of nodes. Additionally [83] and [84] present various techniques that leverage fog-based architectures to improve resiliency over cloud computing setups. Finally [85] outlines a mechanism to enhance the reliability of data transfers in fog-based healthcare networks by using directed diffusion and limited flooding techniques.

2.7 Open Problems and Challenges

In summary, fog computing presents an ideal framework from which to deploy NFV-based services to improve support for delay- and context-sensitive user applications. However there is a dearth of work in this topic area as most SFC embedding studies have focused on larger cloud computing networks and lack critical provisions for resource constraints or end-to-end delays. In light of the above, a range of solutions need to be developed for the fog domain. Foremost there is a need for specialized, scalable SFC embedding algorithms that take into

account key end-user requirements for delay- and context-sensitive support, i.e., including bandwidth throughput, end-to-end delay, and function dependency. Given the reduced size/scale of fog nodes, operator concerns relating to resource constraints and utilization also need to be addressed here. Finally there is a need to design effective survivability schemes for fog-based SFC services in order to meet the reliability needs of high-end users. Specifically several strategies need to be investigated here, including pre- and post-fault recovery mechanisms. Overall these many challenges motivate this research dissertation effort.

Table 2.1: Regular (non-survivability) service demands

	Ref	Preventive	Proactive	Tolerance (Redundancy, Replication)	Node Failure	Link Failure	Single Failure	Multi Failure
Cloud	[58]	✘	✓	✓	✓	✓	✓	✓
	[59]	✓	✓	✓	✓	✘	✓	✘
	[60]	✓	✘	✓	✓	✘	✓	✓
	[61]	✓	✘	✓	✘	✓	✓	✘
	[62]	✓	✘	✓	✓	✓	✓	✘
	[63]	✓	✘	✓	✓	✘	✓	✓
	[64]	✓	✘	✓	✓	✘	✓	✘
	[65]	✓	✓	✓	✘	✓	✓	✓
	[66]	✘	✓	✓	✓	✘	✓	✘
	[67]	✘	✓	✓	✓	✓	✓	✓
	[68]	✘	✓	✓	✘	✓	✓	✘
	[69]	✓	✘	✘	✘	✓	✓	✓
	[70]	✓	✘	✘	✓	✓	✓	✓
	[71]	✘	✓	✘	✓	✓	✓	✓
	[72]	✓	✘	✓	✓	✓	✓	✘
	[73]	✘	✓	✘	✓	✘	✓	✘
Fog	[74]	✓	✓	✘	✓	✓	✓	✓
	[75]	✓	✓	✘	✓	✘	✓	✓
	[76]	✘	✓	✘	✓	✘	✓	✓
	[77]	✓	✓	✘	✓	✓	✓	✘
	[78]	✘	✓	✓	✓	✘	✓	✘
	[79]	✓	✓	✓	✓	✓	✓	✓
	[80]	✘	✓	✓	✓	✘	✓	✓
	[81]	✓	✓	✓	✓	✘	✓	✘
	[82]	✓	✓	✓	✓	✘	✓	✓
	[83]	✘	✓	✓	✓	✘	✓	✓
	[84]	✘	✓	✘	✓	✘	✓	✓
	[85]	✓	✓	✓	✓	✓	✓	✓

Table 2.2: Survivability service demands

	Ref	NFV	SFC	Delay Bound	Lifetime	Prop delay	Queue delay	Proc delay	Trans delay	Network Resources	Energy	Cost
Cloud	[58]	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓
	[59]	✓	✓	✓	✘	✘	✘	✓	✓	✓	✘	✘
	[60]	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
	[61]	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
	[62]	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
	[63]	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓
	[64]	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓
	[65]	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
	[66]	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓
	[67]	✓	✓	✘	✘	✘	✘	✓	✘	✓	✘	✘
	[68]	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
	[69]	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✓
	[70]	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✓
	[71]	✓	✘	✓	✘	✘	✘	✓	✓	✓	✘	✘
	[72]	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
	[73]	✓	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘
Fog	[74]	✓	✓	✘	✘	✘	✘	✓	✓	✓	✘	✘
	[75]	✓	✓	✘	✘	✓	✘	✓	✓	✓	✓	✘
	[76]	✓	✓	✘	✘	✘	✘	✓	✓	✓	✘	✘
	[77]	✓	✓	✘	✘	✘	✘	✓	✓	✘	✘	✘
	[78]	✘	✘	✘	✘	✓	✘	✘	✘	✘	✓	✘
	[79]	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘
	[80]	✘	✘	✓	✓	✘	✓	✓	✓	✘	✘	✓
	[81]	✘	✘	✘	✘	✘	✘	✓	✓	✓	✘	✓
	[82]	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
	[83]	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
	[84]	✘	✘	✘	✘	✓	✘	✓	✓	✘	✘	✘
	[85]	✘	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘

Table 2.3: Survivability requirements and initiative

	Ref	Redundancy	Protection	Scalability	Decentralization	Self-Healing	Self-Diagnose	Recovery Time
cloud	[58]	✓	✘	✓	✓	✘	✘	✘
	[59]	✓	✓	✓	✘	✘	✘	✓
	[60]	✓	✓	✓	✓	✘	✘	✓
	[61]	✓	✓	✓	✓	✓	✘	✓
	[62]	✓	✓	✘	✓	✘	✘	✘
	[63]	✓	✓	✓	✓	✘	✘	✘
	[64]	✓	✓	✘	✓	✘	✘	✓
	[65]	✓	✘	✘	✓	✘	✘	✓
	[66]	✓	✓	✓	✓	✘	✘	✓
	[67]	✓	✓	✓	✓	✘	✘	✘
	[68]	✓	✓	✓	✘	✘	✘	✓
	[69]	✘	✘	✓	✓	✘	✘	✓
	[70]	✘	✘	✓	✓	✘	✓	✓
	[71]	✓	✓	✓	✓	✓	✓	✓
	[72]	✓	✓	✓	✘	✓	✓	✘
	[73]	✘	✘	✘	✘	✓	✓	✓
fog	[74]	✘	✘	✘	✘	✘	✘	✘
	[75]	✘	✘	✓	✓	✘	✘	✓
	[76]	✘	✓	✓	✘	✓	✘	✘
	[77]	✓	✓	✓	✘	✘	✘	✓
	[78]	✓	✘	✓	✓	✘	✘	✓
	[79]	✘	✘	✓	✘	✘	✘	✓
	[80]	✓	✘	✓	✘	✘	✘	✓
	[81]	✓	✓	✓	✓	✘	✘	✓
	[82]	✘	✘	✘	✘	✘	✘	✓
	[83]	✓	✓	✓	✓	✓	✓	✓
	[84]	✘	✘	✓	✓	✘	✘	✓
	[85]	✘	✓	✓	✓	✓	✓	✓

Chapter 3: SFC Provisioning in Fog-Cloud Networks

As detailed in Section 2.7, provisioning VNF service chains across fog networks is a topic of growing interest with relatively little contributions to date. Particularly, what is needed are new SFC embedding methodologies that take into account key requirements, both from the network operator and end-user client sides. Specifically the former include the specifics of fog-based architectures and their associated resource constraints, whereas the latter include important service and QoS parameters. Furthermore the proposed solutions should also be scalable and adaptable to realistic operation settings where demands can either arrive individually (in an on-line manner) or in larger batch sets.

To address these challenges this chapter details a multi-layer architecture for hybrid fog-cloud networks along with a comprehensive SFC demand model that takes into account both end-user bandwidth and delay requirements. Subsequently a novel on-line solution framework is proposed to provision incoming requests in an on-line manner. Specifically two polynomial-time heuristic algorithms are introduced that account for end-user client and operator provisioning concerns, i.e., including methods to minimize delay and load. In addition input SFC demand sorting strategies are also presented to improve provisioning success across a larger batch of incoming demands. Detailed simulation results are then presented and discussed to evaluate the relative performance of the proposed fog SFC mapping and demand sorting strategies.

3.1 Architecture and Notation Overview

Fog computing infrastructures typically consist of multiple layers, as per [35],[48],[50]. Accordingly a generalized multi-layer fog setup is assumed here, as shown in Figure 3.1, and includes the following entities:

- Layer 1 (Terminals): Client terminals including mobile phones, laptops, vehicles, and IoT sensors. End user clients request processing and storage services from the fog network by sourcing SFC requests.

- Layer 2 (Primary Fog): Lower fog layer comprising of smaller resource-constrained nodes connecting to client terminals. Each primary fog node manages a footprint with a set of terminals and also interconnects with its peers.

- Layer 3 (Secondary Fog): Upper fog layer with higher capacity nodes (than Layer 2). Each secondary fog node manages a "cluster" of smaller primary fog nodes and interconnects with its peers. Although the geographic separation between secondary fog nodes is larger, these nodes can still support some delay-sensitive services.

- Layer 4 (Cloud Core): Ultra-scalable cloud datacenters connected to secondary fog nodes over high-bandwidth metro/wide-area networks and links. Given the increased propagation delays involved, this layer is not suitable for highly delay-sensitive services.

Carefully note that all links in the above architecture are assumed to be *wireline* except for those between Layer 1 (terminals) and Layer 2 (primary fog nodes). Furthermore, Layers 2-4 may also contain other networking devices such as routers, gateways, switches, access points, etc (which are not modeled here). Next, consider the requisite notation.

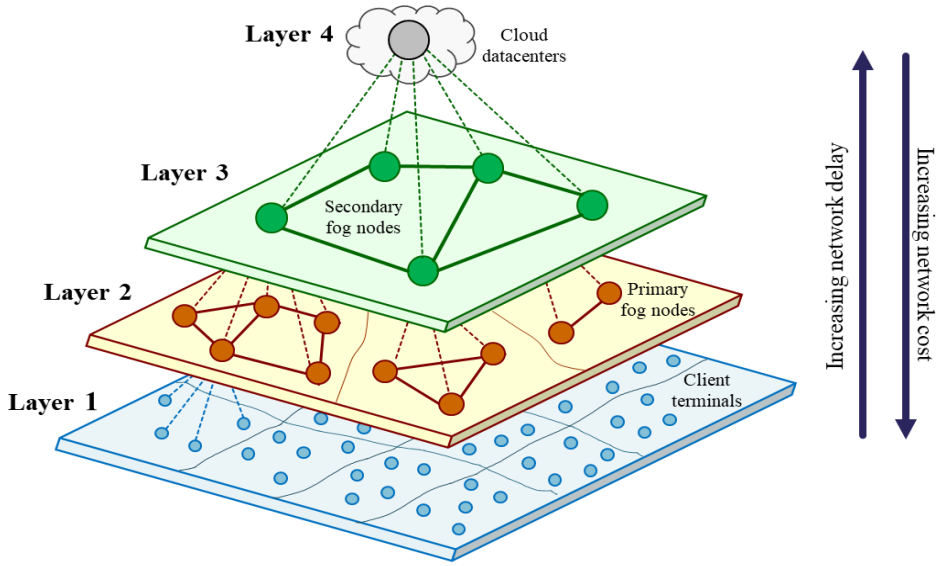

Figure 3.1: Multi-layer fog network architecture

3.1.1 Physical Network Model

The fog-cloud network is modeled as a multi-layer graph, $\mathbf{G}=(\mathbf{N},\mathbf{E})$, where $\mathbf{N}$ is the set of nodes and $\mathbf{E}$ is the set of communication links. Namely $\mathbf{N}=\{n_j^i\}$, where n_j^i is the j-th node at Layer i, and these nodes can either be terminals (i=1) or fog nodes (i=2,3). Since the focus here is on delay-sensitive SFC mapping, cloud core (Layer 4) nodes are not considered. Also, the maximum resource capacity of a fog node in Layer 2 or 3 is given by C_j^i and its free *available* capacity is given by c_j^i ($c_j^i \leq C_j^i$). Similarly, $\mathbf{E}=\{e_{kl}^{ij}\}$, where e_{kl}^{ij} is the link between the k-th node in Layer i and l-th node in Layer j. Here the maximum bandwidth of link e_{kl}^{ij} is B_{kl}^{ij}, its available/free capacity is b_{kl}^{ij} ($b_{kl}^{ij} \leq B_{kl}^{ij}$), and its communication delay is δ_{kl}^{ij} (which includes both propagation and access protocol delays).

Furthermore there are a total of T VNF types denoted by the set $\boldsymbol{F}=\{f_m\}$, where each f_m requires P_m resources at a mapped node. The processing delay for running VNF f_m at node n_j^i is also given by Δ_{ij}^m (i=2-4). Additionally the unit resource usage cost at node n_j^i is χ_j^i, and the unit bandwidth usage cost at link e_{kl}^{ij} is Γ_{kl}^{ij}. As expected upper layers offer increasing node capacity and link bandwidth as well as reduced processing/transmission

delays and resource usage costs. However the increased geographic spread between upper layer nodes entails notably longer communication (propagation) delays, i.e., up to tens of milliseconds (to Layer 4 cloud core datacenters).

3.1.2 SFC Demand Model

The SFC demand model takes into account a range of important end-user client request parameters. Namely, a SFC request, $\mathbf{x}_r$, is summarized via the tuple:

$$\mathbf{x}_r =< src_r, dst_r, b_r, \delta_r, \mathbf{F}_r > \tag{3.1}$$

where src_r is the closest primary fog node to requesting Layer 1 source terminal, dst_r is the closest primary fog node to the Layer 1 destination terminal, b_r is the required bandwidth to interconnect the VNFs, δ_r is the end-to-end delay bound, and $\mathbf{F}_r$ is an *ordered* set of requested functions in the service chain, i.e., $\mathbf{F}_r = \{f_1, f_2, \ldots\} \subseteq \mathbf{F}$. The complete batch set of input SFC requests is also given by the set $\mathbf{X}_{in} = \{\mathbf{x}_r\}$. Accordingly Figure 3.2 illustrates a sample SFC demand request with 4 VNFs being embedded across multiple fog layers, i.e., colors denoting differing VNF types (f_1 - f_4) and the dashed green line denoting the end-to-end SFC connection route (with b_r capacity).

3.2 Polynomial-Time SFC Mapping Schemes

A novel polynomial-time heuristic framework is now presented for SFC embedding in hybrid fog-cloud infrastructures. This solution implements two key schemes for path routing and node mapping across the layers, i.e., *delay* or *load* minimization. The detailed algorithmic psuedocode is shown in Figure 3.3. The scheme starts by constructing a temporary *feasible* graph, $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$, by pruning all nodes and links with insufficient resources to map the SFC request, i.e., nodes with $c^i_j \leq P_m$ ($f_m \in \mathbf{F}_r$) and links with $b^{ij}_{kl} \leq b_r$. Next the K shortest

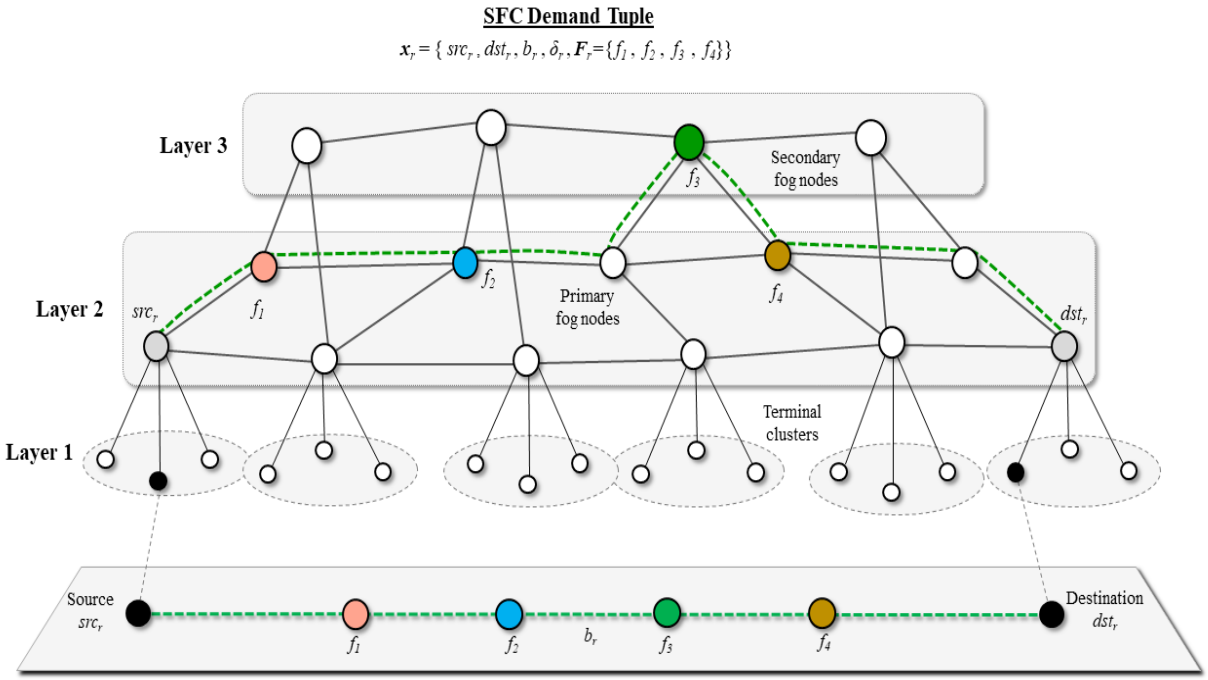

Figure 3.2: SFC demand request embedding example

paths between src_r and dst_r are computed over $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$ with the requirement that they satisfy the delay bound, δ_r. These paths are defined by a set of path vectors, $\mathbf{P}=\{\mathbf{p}_s\}$, each containing vectors of path nodes and links, i.e., $\mathbf{p}_s=\{\{n_j^i\}, \{e_{kl}^{ij}\}\}$. These K path routes are then sorted and searched to find the appropriate VNF mappings for a given provisioning strategy, either delay or cost minimization. These schemes are detailed next.

3.2.1 Delay Minimization Approach (DM)

The *delay minimization* (DM) scheme tries to embed SFC requests to achieve the lowest delay. Accordingly the set of shortest paths, $\mathbf{P}$, is sorted by increasing end-to-end path communication delay, given by the sum of all link delays (transmission, propagation) in a

path vector, i.e.,

$$D_{com}(\mathbf{p}_s) = \sum_{e_{kl}^{ij} \in \mathbf{p}_s} \delta_{kl}^{ij}, \text{where } n_k^i, n_l^j \in \mathbf{p}_s \tag{3.2}$$

Next the individual paths, $\mathbf{p}_s$, are searched to find mappings for the VNFs in $\mathbf{F}_r$ with the lowest processing delay (outer loop, Figure 3.3). Now realistically the number of nodes along a path will be greater than the number of requested VNFs and this gives up to $Q=\binom{|\mathbf{p}_s|}{|\mathbf{F}_r|}$ mapping combinations, where $|\cdot|$ denotes set cardinality. Hence the algorithm sequentially evaluates all such combinations (inner loop, Figure 3.3) by calling an appropriate combination generating function. Namely, $\Phi(|\mathbf{p}_s|, |\mathbf{F}_r|, t)$ returns the t-th combination vector of $\binom{|\mathbf{p}_s|}{|\mathbf{F}_r|}$, $\mathbf{u}_t$, of length $|\mathbf{F}_r|$. Specifically, $\mathbf{u}_t=\{u_t^m\}$, where u_t^m notes the position of the m-th VNF mapping, $1 \leq u_t^m \leq |\mathbf{p}_s|$, where $|\mathbf{p}_s|$ is the number of nodes in $\mathbf{p}_s$. Based upon the above, the combination vector $\mathbf{u}_t$ is used to build a SFC *mapping vector* of 2-tuple pairs, $\mathbf{v}_t=\{<f_m, n_j^i>\}$, noting the mapped nodes for each VNF $f_m \in \mathbf{F}_r$, Figure 3.3. Now a VNF mapping also imposes processing delays at the mapped nodes, given by:

$$D_{prc}(\mathbf{v}_t) = \sum_{n_j^i \in \mathbf{v}_t} \Delta_{ij}^m, \text{where } \mathbf{v}_t = \{< f_m, n_j^i >\} \tag{3.3}$$

Hence the inner loop in Figure 3.3 searches all mapping combinations to find the one with the lowest aggregate delay, given by $D_{com}(\mathbf{p}_s)+D_{prc}(\mathbf{v}_t)$. Overall the DM scheme favors closer primary fog nodes in Layer 2 with lower delays/higher cost (although secondary fog nodes will also be considered and used if necessary). Expectedly, an exhaustive search over all Q combinations can be problematic for longer paths with few VNFs. Hence a faster *first fit-DM* (FF-DM) mapping is also defined to select the first valid mapping (Figure 3.3).

3.2.2 Load Minimization Approach (LM)

Meanwhile the *load minimization* (LB) scheme tries to minimize the *relative* load at network links and nodes when embedding SFC requests (while also satisfying the delay bound, δ_r). Hence the K shortest-paths, $\mathbf{P}$, are now sorted in order of increasing aggregate link load, defined as:

$$L_{link}(\mathbf{p}_s) = \sum_{e^{ij}_{kl} \in \mathbf{p}_s} \Big(\frac{B^{ij}_{kl}}{b^{ij}_{kl} + \epsilon} \Big), \text{where } n^i_k, n^j_l \in \mathbf{p}_s \tag{3.4}$$

where $\epsilon \ll 1$ is chosen to avoid divide-by-zero errors. Namely the fractional term in Eq. 3.4 defines the relative load of a link. Specifically this value is equal to unity if the link capacity is unused (b^{ij}_{kl}=B^{ij}_{kl}) but approaches infinity if it is fully reserved (b^{ij}_{kl}=0). These sorted paths are then searched to find the VNF mapping with the lowest aggregate node load (outer loop, Figure 3.3). Similarly, a node-level load is also defined for a mapping vector $\mathbf{v}_t$ as follows:

$$L_{node}(\mathbf{v}_t) = \sum_{n^i_j \in \mathbf{v}_t} \Big(\frac{C^i_j}{c^i_j + \epsilon} \Big) \tag{3.5}$$

where the inner bracketed term is the relative load of a node (akin to Eq. 3.4). As per the above all mapping combinations are evaluated (inner loop, Figure 3.3) to find a feasible one with the lowest aggregate load, i.e., $L_{link}(\mathbf{p}_s) + L_{node}(\mathbf{v}_t)$. Overall the LB approach avoids closer Layer 2 fog nodes if they are more congested and instead maps more VNFs onto Layer 3 nodes (lower cost). A faster *first fit-LM* (FF-LM) mapping scheme is also defined to select the first valid load-based mapping (Figure 3.3).

3.3 Algorithmic Complexity

Consider the run-time complexity of the algorithm in Figure 3.3. First, assuming efficient implementation, K shortest-path route computation over $\mathbf{G}(\mathbf{N}, \mathbf{E})$ is bounded by

$O(K|\mathbf{N}|\log(|\mathbf{E}|))$. Meanwhile the main loop in Figure 3.3 iterates over the K paths and examines all M combinations by performing linear computations for each (to find lowest path delay or load). Assuming path lengths bounded by $L \ll |\mathbf{N}|$, the number of combinations is bounded by $O(L^L)$. Hence the total complexity of the scheme is given by $O(K|\mathbf{N}|\log(|\mathbf{E}|) + K \cdot L^{L+1})$. This bound also applies to the first choice schemes (FF-DM, FF-LM).

3.4 Batch Sorting Strategies

In many cases, operators may process user demands in batches at fixed intervals of time (aperiodic, periodic). In particular a batch represents set of multiple requests and can be of variable size. Overall, batch processing can help streamline provisioning transactions and also enable more efficient and effective resource allocation versus "on-demand" provisioning, i.e., where incoming SFC demands are mapped in the order in which they are received. Namely, considering a large number of demands in conjunction with each other allows operators to design/apply more expansive provisioning methodologies, e.g., such as those based upon optimization, metaheuristics, or sorting/re-ordering algorithms.

In light of the above several demand sorting strategies are proposed here to re-order incoming SFC batch requests according to various criteria. The overall goal here is to try to lower SFC request blocking rates (i.e., increase revenues) and/or cost. Now as noted earlier in Section 3.1.2, the complete batch of input SFC requests is given by the set $\mathbf{X}_{in} = \{\mathbf{x}_r\}$. Accordingly the proposed sorting strategies transform/re-order this input set, i.e., $\mathbf{X}_{in} \longrightarrow \mathbf{X}'$, based upon their bandwidth and VNF resource request sizes. As per well-known bounds on optimized sorting algorithms, all of these schemes have a run-time computational complexity of $O(|\mathbf{X}_{in}|log(|\mathbf{X}_{in}|))$. Consider the details.

3.4.1 Highest Bandwidth First (HBF)

This scheme sorts batch SFC requests in order of decreasing bandwidth size, i.e., maps larger b_r values first:

$$\mathbf{X}_{in} \longrightarrow \mathbf{X}^{'} = \{\mathbf{x}_r \mid b_r \geq b_{r+1}\} \tag{3.6}$$

The overall objective here is to provision higher revenue demands first as customers with larger bandwidth sizes will likely provide increased revenues.

3.4.2 Lowest Bandwidth First (LBF)

This scheme sorts batch SFC requests in order of increasing bandwidth size, i.e., maps smaller b_r values first:

$$\mathbf{X}_{in} \longrightarrow \mathbf{X}^{'} = \{\mathbf{x}_r \mid b_r \leq b_{r+1}\} \tag{3.7}$$

The overall objective here is to minimize bandwidth fragmentation (wastage) on network links by packing in smaller demands first.

3.4.3 Highest VNF First (HVF)

This scheme sorts batch SFC requests in order of decreasing average VNF resource size, i.e., maps larger averages first:

$$\mathbf{X}_{in} \longrightarrow \mathbf{X}^{'} = \{\mathbf{x}_r \mid \sum_{f_m \in \mathbf{F}_r} \frac{f_m}{|\mathbf{F}_r|} \geq \sum_{f_m \in \mathbf{F}_{r+1}} \frac{f_m}{|\mathbf{F}_{r+1}|}\} \tag{3.8}$$

The overall objective here is to provision higher revenue demands first as customers with larger VNF resource demand will likely provide increased revenues.

Table 3.1: Multi-layer fog network simulation parameters

Parameter	Layer 2 Primary Fog	Layer 3 Secondary Fog
Number of nodes	20 (5 clusters)	5 (one/cluster)
Node resource capacity, C_j^i	20	200
Node VNF proc. delay, Δ_{ij}^m (ms)	1	0.5
Node unit usage cost, χ_j^i	5	1
Number of links	200 (40/cluster)	10 links
Link bandwidth, B_{kl}^{ij}	100	1,000
Link delay, δ_{kl}^{ij} (ms)	0.5-1 (uniform)	1-2 (uniform)
Bandwidth unit usage cost, Γ_{kl}^{ij}	5	1

Parameter	SFC Request
Total number of VNFs, T	6
VNFs per SFC request, $\lvert\mathbf{F}_r\rvert$	4-6 (uniform)
Node resources per VNF, P_m	2-3 (uniform)
SFC bandwidth request, b_r	4-6 (uniform)
SFC delay bound, δ_r (ms)	25

3.4.4 Lowest VNF First (LVF)

This scheme sorts batch SFC requests in order of increasing average VNF resource size, i.e., maps smaller averages first:

$$\mathbf{X}_{in} \longrightarrow \mathbf{X}' = \{\mathbf{x}_r \mid \sum_{f_m \in \mathbf{F}_r} \frac{f_m}{|\mathbf{F}_r|} \leq \sum_{f_m \in \mathbf{F}_{r+1}} \frac{f_m}{|\mathbf{F}_{r+1}|}\} \tag{3.9}$$

The overall objective here is to minimize node resource fragmentation (wastage) at network nodes by packing in smaller demands first.

3.5 Performance Analysis

The SFC mapping schemes are evaluated for a realistic multi-layer fog network with 1,000 terminals (Layer 1), 100 primary fog nodes (Layer 2), and 5 secondary fog nodes (Layer 3), Table 3.1. Each secondary fog node manages a "cluster" of 20 primary fog nodes, and some

Input: Network $\mathbf{G}$=($\mathbf{N}$, $\mathbf{E}$), request $\mathbf{x}_r$=<src_r, dst_r, b_r, δ_r, $\mathbf{F}_r$>
Output: SFC node mapping, $\mathbf{v}^*$, and SFC route, $\mathbf{p}^*$
/ Prune network to build feasible graph, $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$ */*
$\mathbf{N}' \leftarrow$ Remove nodes $n_j^i \in \mathbf{N}$ with $c_j^i < \min\{f_m \in \mathbf{F}_r\}$ & $\mathbf{E}' \leftarrow$ Remove links $e_{kl}^{ij} \in \boldsymbol{E}$ with $b_{kl}^{ij} < b_r$
Compute K shortest (delay-constrained) paths from $src_r^{'}$ to $dst_r^{'}$ in $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$, i.e., $\mathbf{P}$=$\{\mathbf{p}_s\}$ s.t. $D_{com}(\mathbf{p}_s) < \delta_r$
/ Order K paths based on scheme */*
if (delay minimization) **then**
    Sort $\mathbf{P}$ by increasing delay, i.e., $D_{com}(\mathbf{p}_s) \leq D_{com}(\mathbf{p}_{s+1})$
else if (load balancing) **then**
    Sort $\mathbf{P}$ by increasing load, i.e., $L_{link}(\mathbf{p}_s) \leq L_{link}(\mathbf{p}_{s+1})$
/ Initialize path vector, mapping vector, and flags */*
$\mathbf{p}^*$=$\{\emptyset\}$, $\mathbf{v}^*$=$\{<\emptyset, \emptyset>\}$, $found$=0
/ Search all K paths, */*
for (each path) **do**
    Extract path $\mathbf{p}_s \in \mathbf{P}$, i.e., path vector
    / Compute max. number of mapping combinations */*
    Q=choose($|\mathbf{p}_s|$,$|\mathbf{F}_r|$)
    / Search all combinations */*
    for (each combination) **do**
        / Generate mapping vector */*
        Compute t-th comb. vector, $\mathbf{u}_t$=$\{u_t^m\}$=$\Phi(|\mathbf{p}_s|, |\mathbf{F}_r|, t)$
        Build t-th node mapping pair vector, $\mathbf{v}_t$=$\{<f_m, n_j^i>\}$
        / Check delay and node resources for mapping */*
        if (($D_{com}(\mathbf{p}_s) + D_{prc}(\mathbf{v}_t) \leq \delta_r$) & ($c_j^i \geq P_m$ for all mapped nodes in $\mathbf{v}_t$)) **then**
            $found$=1
            / Track & update latest path and mapping vectors */*
            if (first fit-DM or first fist-LB) **then**
                Copy $\mathbf{p}_s \rightarrow \mathbf{p}^*$, $\mathbf{v}_t \rightarrow \mathbf{v}^*$
                Exit search */* first valid delay/load-based mapping */*
            else if (delay minimization) **then**
                if ($D_{com}(\mathbf{p}_s)$+$D_{prc}(\mathbf{v}_t)$<$D_{com}(\mathbf{p}^*)$+$D_{prc}(\mathbf{v}^*)$) **then**
                    Copy $\mathbf{p}_s \rightarrow \mathbf{p}^*$, $\mathbf{v}_t \rightarrow \mathbf{v}^*$
            else if (load balancing) **then**
                if ($L_{link}(\mathbf{p}_s)$+$L_{node}(\mathbf{v}_t)$<$L_{link}(\mathbf{p}^*)$+$L_{node}(\mathbf{v}^*)$) **then**
                    Copy $\mathbf{p}_s \rightarrow \mathbf{p}^*$, $\mathbf{v}_t \rightarrow \mathbf{v}^*$
/ Reserve resources on path $\mathbf{p}^*$ for mapping $\mathbf{v}^*$ */*
if ($found$) **then**
    Reserve node capacity in $\mathbf{N}$ (i.e., decrement available capacity of all mapped nodes in $\mathbf{v}^*$ by P_m values for $f_m \in \mathbf{F}_r$)
    Reserve link capacity in $\mathbf{E}$ (decrement available capacity of all mapped links in $\mathbf{p}^*$ by b_r)

Figure 3.3: Fog-based SFC mapping algorithm for a request, $\mathbf{x}_r$

primary fog nodes are connected to multiple secondary fog nodes. In turn each primary fog node manages 5 terminals, and respective node resource (link bandwidth) levels are scaled up between layers, Table 3.1. A total of T=6 VNFs are defined, and SFC requests are generated between random terminals. Each SFC request has 4-6 VNFs and requires 4-6 bandwidth units (b_r). To represent a range of functions, each VNF requires 2-3 units of node resource (P_m) and has 0.5-2 ms node processing delay (Δ_{ij}^m). The delay bound (δ_r) is also set to 25 ms to reflect delay sensitive users. Finally the number of computed shortest-paths is set to K=3 (Figure 3.3) as results show minimal gains with larger values. Tests are done for varying SFC demand batches (infinite holding times) and several metrics are evaluated.

3.5.1 SFC Mapping Results

First, Figure 3.4 plots the SFC request blocking rates for all schemes. This metric is typically regarded as a measure of revenue for network operators. These results indicate that the LM scheme gives the highest carried load as mappings are spread across Layer 2 (and Layer 3) nodes to prevent network hot spots (resource exhaust). By contrast the DM schemes (DM, FF-DM) strictly focus on shortest delay routes and tend to concentrate node mappings onto Layer 2 nodes (primary fog nodes). The net result here is notably higher blocking rates. For example the exhaustive-search DM scheme consistently yields 10-15% lower success rates than the LM scheme for all input batch sizes tested (and even about 5% lower than the FF-LM scheme).

Next the average end-to-end delays for successful SFC demands are plotted in Figure 3.5. These values include processing, transmission, and propagation times. As expected the DM scheme gives the lowest delay due to mapping preference onto closer primary fog nodes. By contrast the LM scheme has notably higher latency since the non-linear weights in Eqs. 3.4 and 3.5 increase rapidly with congestion at Layer 2 nodes and links. Regardless these increased delays still fall within the 25 ms bound and also pertain to a larger number

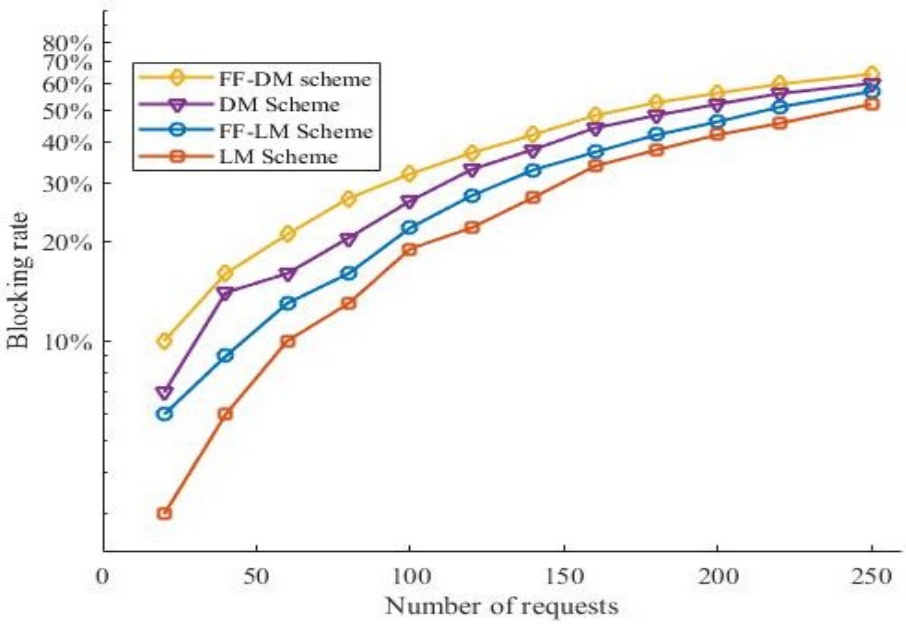

Figure 3.4: SFC demand request blocking rate

of accepted requests, Figure 3.4. Note that the LM scheme also gives lower delays versus its simpler FF-LM variant, i.e., by about 15-20% at low-medium loads, indicating a clear benefit with expanded search.

Finally the average embedding cost for all successful SFC demands is also gauged in Figure 3.6. Specifically, the cost of a mapped request, $\mathbf{x}_r$, is computed as the total cost of all the node and link resources that it uses, i.e.,

$$cost(\mathbf{x}_r) = \sum_{n_j^i \in \mathbf{v}^*, f_m \in \mathbf{F}_r} \chi_j^i \cdot P_m + \sum_{e_{kl}^{ij} \in \mathbf{p}^*} \Gamma_{kl}^{ij} \cdot b_r \tag{3.10}$$

Overall results indicate that the LM scheme gives the lowest cost as it makes extensive use of Layer 3 resources. Conversely, provisioning SFC demands with the DM scheme is much more expensive, with costs ranging from 130% (low load) to 45% (high load) above the LM scheme. In fact these findings also confirm that both first-fit strategies outperform the DM

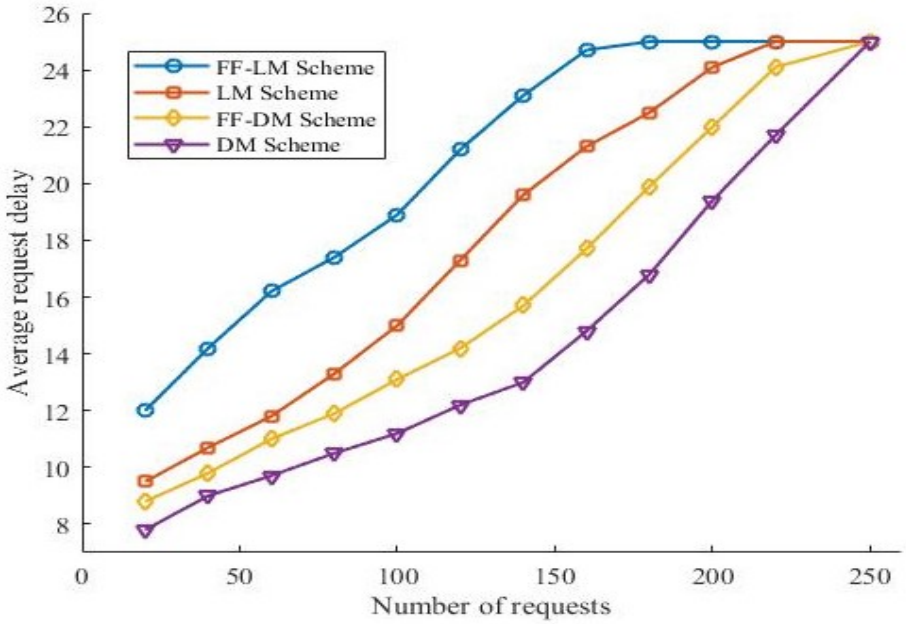

Figure 3.5: Average end-to-end SFC delay (successful requests)

scheme, including the FF-DM scheme. As such the LM strategy will likely yield improved revenue profiles for network operators.

3.5.2 Batch Sorting Results

The effectiveness of the various batch sorting schemes in Section 3.4 is now analyzed for varying input demand sets. First consider delay-based provisioning. Namely Figure 3.7 plots the impact of all four sorting strategies on the blocking performance of the DM scheme (as well as the non-sorted baseline approach). As expected the DM-LVF and DM-LBF schemes give the lowest blocking as they quickly pack in a larger number of smaller/lighter demands first. By contrast the HBF and HVF schemes yield several factors higher blocking, even exceeding the non-sorted baseline scheme. For example at a nominal input load of 50 requests, the DM-LBF and DM-LVF schemes give approximately 5% and 7% blocking, respectively, whereas the DM-HBF and DM-HVF schemes yield almost 5 times higher blocking. This

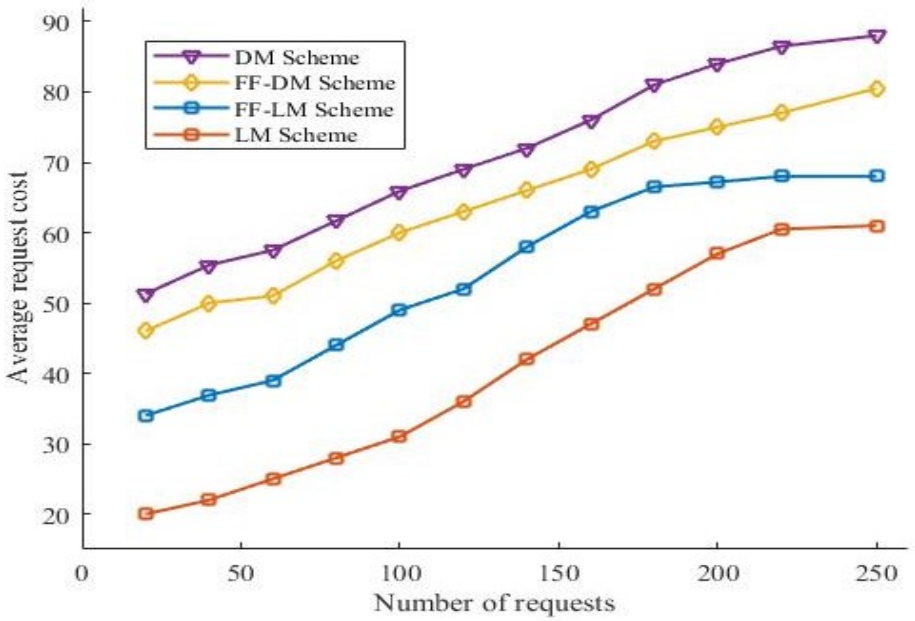

Figure 3.6: Average SFC cost (successful requests)

increased blocking is a direct result of the increased (node, link) resource fragmentation that occurs when provisioning larger demands first.

Similar runs are also repeated for the load balancing LM scheme, and the results in Figure 3.8 also yield very similar findings for the different sorting strategies. Namely the LM-LBF and LM-LVF schemes give superior blocking performance, coming in several factors lower than their LM-HBF and LM-HVF counterparts for low-medium input load ranges (request batch sizes). In line with the findings in Section 3.5.1, the LM scheme also outperforms the DM scheme for each sorting strategy. For example comparing the results for LVF sorting between Figures 3.7 and 3.8 shows almost 80-100% lower blocking rates for up to medium input loads (150 requests). Again this notable improvement is due the ability of the LM scheme to spread demands across higher capacity Layer 3 (secondary fog) nodes.

Next the average end-to-end delay is also analyzed for the respective sorting strategies, i.e., HBF, HVF, LVF, LBF. Foremost, results for the DM approach in Figure 3.9 indicate that the least resource first strategies give notably lower delays across all input ranges. In

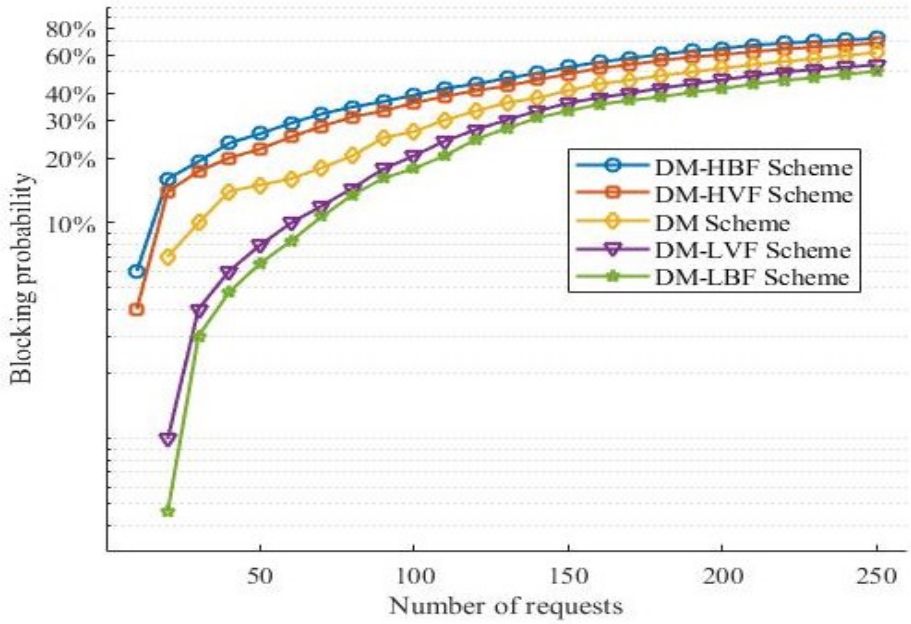

Figure 3.7: SFC demand request blocking rate, DM sorting scheme

fact the respective delays for both DM-LBF and DM-LVF schemes stay well below the 25 ms bound except at extremely high input loads (over 250 requests). Overall, preferentially routing smaller demands first gives more SFC mappings onto lower Layer 2 nodes/links, helping lower average end-to-end delays. By contrast the HBF and HVF strategies process heavier demands first, quickly exhausting resources at the primary fog layer (increasing node processing delays) and pushing the lighter demands onto more distant secondary fog layer nodes (increasing propagation delays). These effects yield notably higher end-to-end SFC delays, i.e., averaging almost 5 ms higher than the corresponding LBF and LVF schemes in Figure 3.9.

Similarly, end-to-end delay results are also presented for the LM scheme in Figure 3.10. As per the earlier findings with the DM scheme, the LBF and LVF sorting strategies also yield the lowest end-to-end delays. However the relative separation between the different sorting strategies is much lower here, i.e., within 5 ms, since the LM scheme is not focused on delay minimization. Furthermore, comparing the results in Figure 3.9 and Figure 3.10 shows that

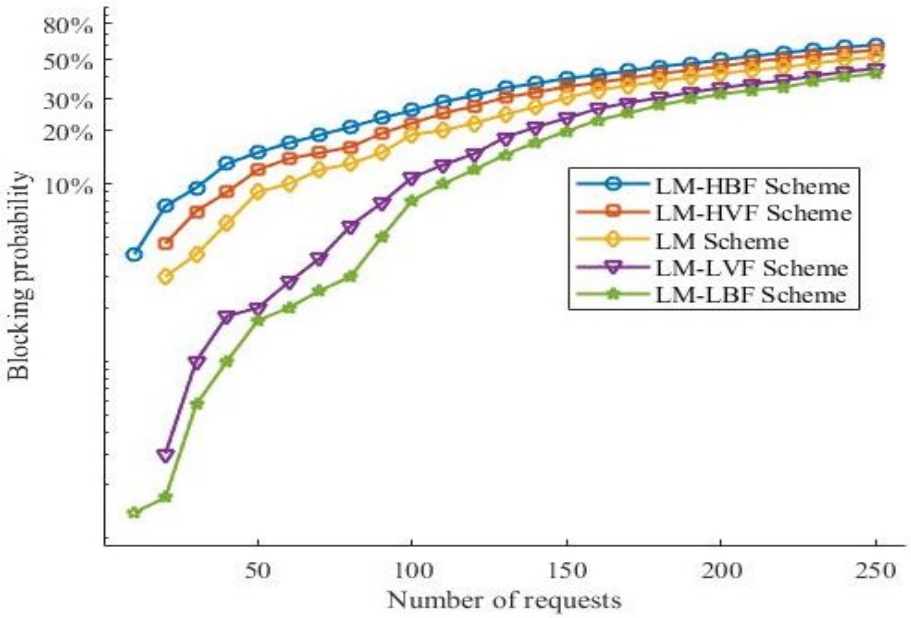

Figure 3.8: SFC demand request blocking rate, LM sorting scheme

the LM sorting variants give notably higher end-to-end delays than their DM counterparts. For example the LM-LBF scheme saturates at the 25 ms delay bound for slightly under 250 requests, whereas the DM-LBF scheme only starts to approach this bound for 300 requests. Overall this is expected since the LM scheme makes more use of Layer 3 secondary fog nodes and links.

Average SFC embedding costs are also analyzed for the different sorting strategies, as per Eq. 3.10. Foremost, results for the DM scheme are shown in Figure 3.11 and indicate that the least resource first schemes, e.g., LBF and LVF, actually yield the highest costs. This is due to the fact that these methods pack a larger number (of smaller) demands onto primary layer nodes/links with higher resource usage costs. In turn this increased utilization drives up average SFC costs. By contrast the highest resource first schemes tend to increase resource fragmentation at Layer 2 nodes, forcing more mappings onto Layer 3 nodes/links with lower costs. Indeed the differences are notable here, with the widest margins occurring between the HBF and LBF sorting strategies, i.e., anywhere from 15-30% higher costs with

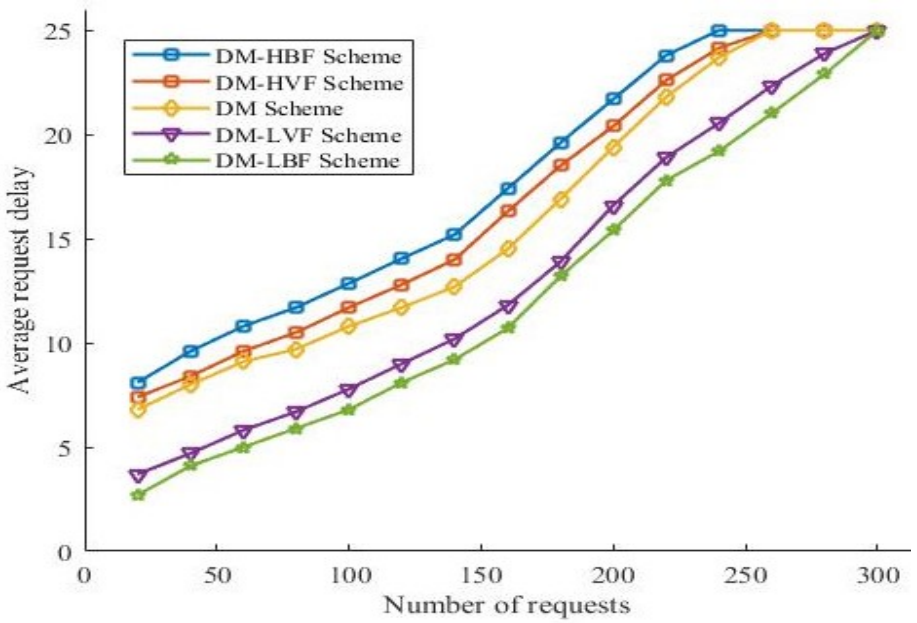

Figure 3.9: Average end-to-end SFC delay, DM sorting scheme (successful requests)

the former sorting strategy. Overall this is the cost penalty for achieving improved delay performances with the LBF and LVF strategies, as per Figures 3.9 and 3.10.

Finally Figure 3.12 plots the average SFC embedding costs for the LM scheme. Again, the relative performance between the different sorting strategies matches the results for the DM scheme in Figure 3.11. Namely the LM-LBF (LM-HBF) scheme gives the highest (lowest) costs, with the LVF strategies falling in between. More importantly, comparing the results in Figures 3.11 and 3.12 also shows notably lower costs with the LM schemes. For example the average SFC cost of the LM-HBF scheme for up to 100 requests is well below 20, i.e., almost 3 times less than the DM-HBF scheme. Although the comparative differences between the other sorting strategies are less drastic between the DM and LM schemes, they are still quite notable, e.g., for 200 input requests the LM-LVF scheme gives an average SFC cost of 62 (Figure 3.12) versus 90 for the DM-LVF scheme (Figure 3.11).

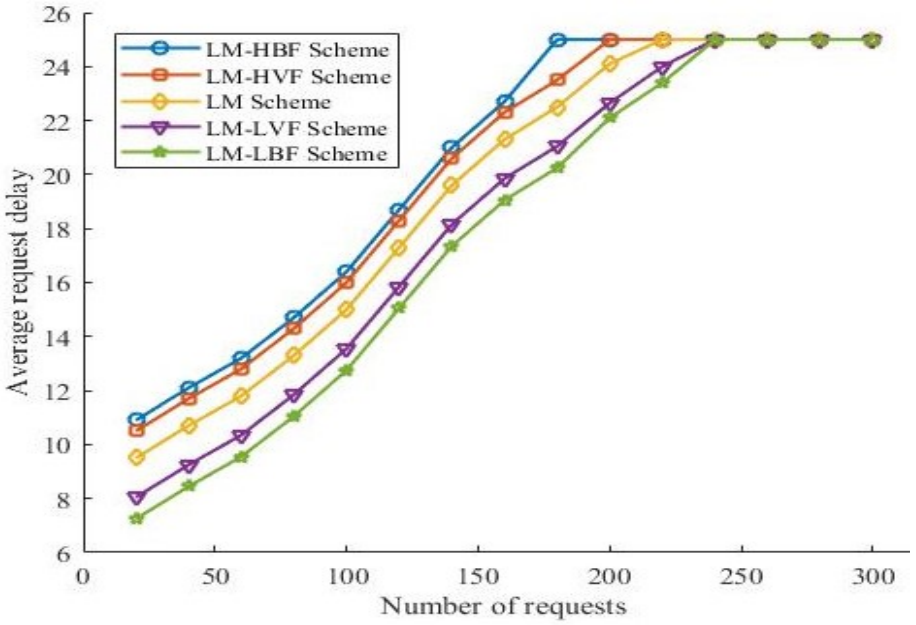

Figure 3.10: Average end-to-end SFC delay, LM sorting scheme (successful requests)

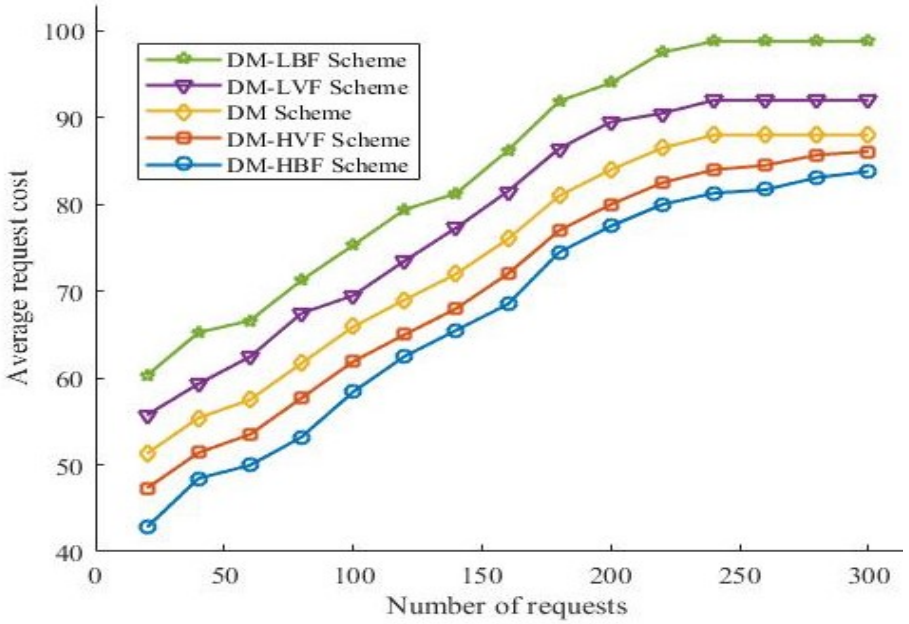

Figure 3.11: Average SFC cost, DM sorting scheme(successful requests)

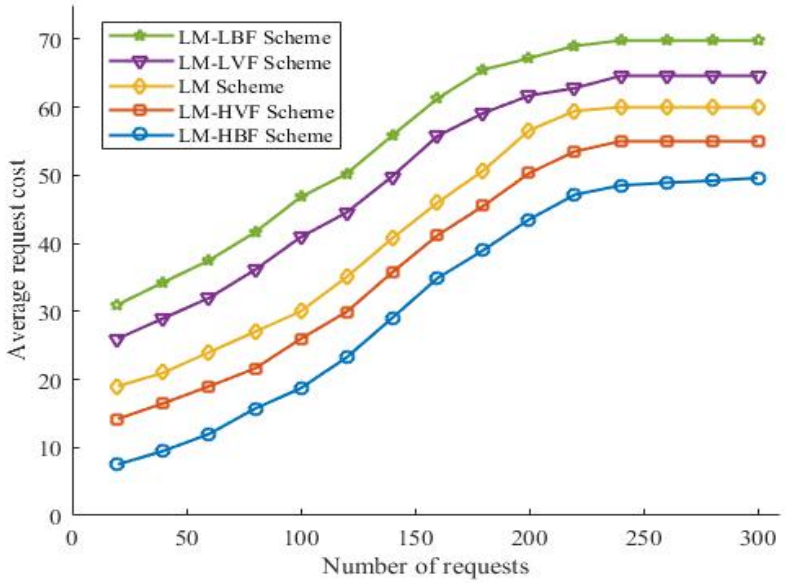

Figure 3.12: Average SFC cost, LM sorting scheme (successful requests)

Chapter 4: Pre-Provisioned SFC Protection

As fog-based SFC services gain traction associated survivability concerns are also starting to emerge. Now as surveyed in Section 2.5, most existing studies on SFC reliability have used protection-based strategies to pre-provision backup VNF or link bandwidth resources for mitigating failures. Although these methods are mostly designed to handle isolated single faults, this is still a critical requirement as these types represent the overwhelming majority of network failures. In light of this there is a further need to develop realistic and effective protection methodologies for SFC demands across fog networks as well.

To address these survivability concerns, this chapter augments the heuristic framework presented in Chapter 3 to support further dedicated pre-provisioned protection capabilities. The focus here is to handle common single node or link faults. Performance results are also presented from a simulation study to evaluate the blocking rate, delay and cost of these proposed SFC protection strategies.

4.1 Polynomial-Time SFC Protection Schemes

Novel polynomial-time heuristic schemes are now presented for SFC survivability in fog networks. Specifically these methods extend upon the non-survivable heuristics developed in Chapter 3 by pre-provisioning *backup* node/link-disjoint SFC mappings. Namely two SFC mappings are now provisioned for each incoming request, $\mathbf{x}_r$, termed as *primary* and *backup*. Furthermore akin to Section 3.2, delay and load minimization strategies are also used for path routing and node mapping across layers. The case of end-to-end SFC protection is also illustrated in Figure 4.1 for the same 4-VNF SFC embedding example presented earlier in

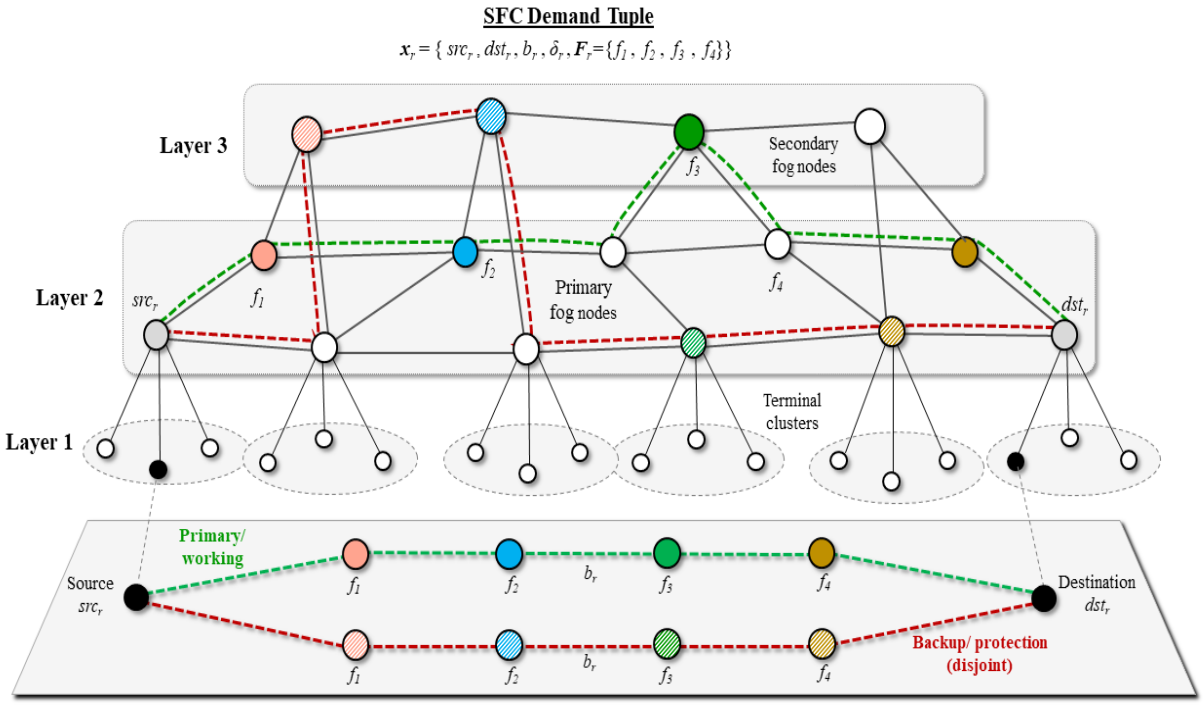

Figure 4.1: SFC demand protection example

Figure 3.2. Here a complete link and node-disjoint backup SFC mapping is established to protect the original primary "working" SFC embedding. As expected this technique entails much more resource overheads at both the nodes and links. Consider the details.

The overall psuedocode for dedicated SFC protection is shown in Fig. 4.2. This framework reuses the notation presented in Section 3.1.2, i.e., where a network is defined as a graph $\mathbf{G}$=($\mathbf{N}, \mathbf{E}$), and a demand is given by a tuple $\mathbf{x}_r$. The algorithm starts out by selecting the closest (Layer 2) primary fog nodes to the source and destination terminals to "anchor" the SFC, i.e., src_r and dst_r. A temporary *feasible* graph, $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$, is then built by pruning all nodes and links with insufficient resources to map the SFC request, i.e., nodes with $c_j^i \leq P_m$ ($f_m \in \mathbf{F}_r$) and links with $b_{kl}^{ij} \leq b_r$. Next, the K shortest paths are computed over $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$ with the requirement that they satisfy the delay bound, δ_r. These paths are

termed as the primary (working) paths and denoted by a set of path vectors, $\mathbf{P}$=$\{\mathbf{p}_s\}$, each containing vectors of path nodes and links, i.e., $\mathbf{p}_s$=$\{\{n_j^i\},\{e_{kl}^{ij}\}\}$. For backup protection SFC provisioning, node/link-disjoint paths are then computed for each path in $\mathbf{p}_s$, denoted by the set $\mathbf{Q}$=$\{\mathbf{q}_s\}$. The node/link disjoint path pairs in $\{\mathbf{P},\mathbf{Q}\}$=$\{\mathbf{p}_s,\mathbf{q}_s\}$ are then iteratively searched to find the appropriate VNF mappings for a given provisioning strategy, e.g., delay or load minimization. Carefully note that these protection heuristics have the same computational complexity as the regular non-survivable schemes presented in Section 3.2, i.e., $O(K|\mathbf{N}|\log(|\mathbf{E}|) + K \cdot L^{L+1})$, where L is the path length, $L \ll |\mathbf{N}|$. Further details are now presented.

4.1.1 Delay Minimization Protection (DM-P) Approach

The *delay minimization-protection* (DM-P) approach focuses on minimizing the total delay across both the primary and backup SFC mappings. To achieve this, the set of K path pairs $\{\mathbf{P},\mathbf{Q}\}$ are sorted by increasing end-to-end communication delay across both the primary and backup routes. Namely this value is given by the sum of all link (transmission, propagation) delays between nodes in both path vectors, i.e.,

$$D_{com}(\mathbf{p}_s, \mathbf{q}_s) = \sum_{e_{kl}^{ij} \in \mathbf{p}_s} \delta_{kl}^{ij} + \sum_{e_{kl}^{ij} \in \mathbf{q}_s} \delta_{kl}^{ij} \tag{4.1}$$

Next the above path pairs are searched to find mappings for the VNFs in $\mathbf{F}_r$ with the lowest processing delay (outer loop, Fig. 4.2). As per the regular provisioning case in Section 3.2, the number of nodes along the primary/backup paths will usually be greater than the number of requested VNFs. Hence this implies up to Q_1=$\binom{|\mathbf{p}_s|}{|\mathbf{F}_r|}$ mapping combinations for the primary working path and Q_2=$\binom{|\mathbf{q}_s|}{|\mathbf{F}_r|}$ combinations for the backup path. Hence the algorithm sequentially evaluates all of these combinations for both paths (inner loop, Fig. 4.2) by calling an appropriate combination generating function. Namely $\Phi(|\mathbf{p}_s|, |\mathbf{F}_r|, t)$ returns the t-th

combination vector of $\binom{|\mathbf{p}_s|}{|\mathbf{F}_r|}$, $\mathbf{u}_t$, of length $|\mathbf{F}_r|$ for the primary path. Specifically, $\mathbf{u}_t$=$\{u_t^m\}$, where u_t^m notes the position of the m-th VNF mapping on the primary path, $1 \leq u_t^m \leq |\mathbf{p}_s|$. The combination vector $\mathbf{u}_t$ is used to build a SFC mapping vector of 2-tuple pairs for the primary path, $\mathbf{v}_t$=$\{<f_m, n_j^i>\}$, to note the mapped nodes for each VNF $f_m \in \mathbf{F}_r$, Fig. 4.2. Similarly $\Phi(|\mathbf{q}_s|, |\mathbf{F}_r|, t)$ returns the t-th combination vector of $\binom{|\mathbf{q}_s|}{|\mathbf{F}_r|}$, $\mathbf{y}_t$, of length $|\mathbf{F}_r|$ for the backup path. Specifically $\mathbf{y}_t$=$\{y_t^m\}$, where y_t^m notes the position of the m-th VNF mapping on the backup path, $1 \leq y_t^m \leq |\mathbf{q}_s|$. Finally $\mathbf{y}_t$ is used to build a SFC mapping vector of 2-tuple pairs for the backup path, $\mathbf{w}_t$=$\{<f_m, n_j^i>\}$, Fig. 4.2. Now the primary and backup VNF mappings also impose processing delays at the mapped nodes, given by:

$$D_{prc}(\mathbf{v}_t, \mathbf{w}_t) = \sum_{n_j^i \in \mathbf{v}_t} \Delta_{ij}^m + \sum_{n_j^i \in \mathbf{w}_t} \Delta_{ij}^m \tag{4.2}$$

Hence the inner loop in Fig. 4.2 searches all mapping combinations to find the one with the lowest aggregate delay, given by $D_{com}(\mathbf{p}_s, \mathbf{q}_s)$+$D_{prc}(\mathbf{v}_t, \mathbf{w}_t)$. Overall the DM scheme will favor closer primary fog nodes in Layer 2 with lower delays/higher cost, although secondary fog nodes in Layer 3 will also be considered and used if necessary. A *first fit-DM-P* (FF-DM-P) scheme is also defined to reduce run-time complexity by selecting the first valid mapping in Fig. 4.2 (as per non-survivable DM scheme, Section 3.2.1).

4.1.2 Load Minimization Protection (LM-P) Approach

The *load minimization-protection* (LM-P) scheme tries to minimize the *relative* network link and node load when provisioning primary/backup SFC mappings (while also satisfying the request delay bound, δ_r). Hence the K primary/backup path pairs {$\mathbf{P}$,$\mathbf{Q}$} are now sorted

in order of increasing aggregate link load, defined as:

$$L_{link}(\mathbf{p}_s, \mathbf{q}_s) = \sum_{e_{kl}^{ij} \in \mathbf{p}_s} \left(\frac{B_{kl}^{ij}}{b_{kl}^{ij} + \epsilon} \right) + \sum_{e_{kl}^{ij} \in \mathbf{q}_s} \left(\frac{B_{kl}^{ij}}{b_{kl}^{ij} + \epsilon} \right) \tag{4.3}$$

where $\epsilon \ll 1$ is chosen to avoid divide-by-zero errors. Namely the fractional term in Eq. (4.3) is the relative link load (as defined in the non-survivable LM scheme, Section 3.2.2). These sorted paths are then searched to find the VNF mapping with the lowest aggregate node load (outer loop, Fig. 4.2). Furthermore a node-level load is also defined for a mapping $\mathbf{v}$ as follows:

$$L_{node}(\mathbf{v}_t, \mathbf{w}_t) = \sum_{n_j^i \in \mathbf{v}_t} \left(\frac{C_j^i}{c_j^i + \epsilon} \right) + \sum_{n_j^i \in \mathbf{w}_t} \left(\frac{C_j^i}{c_j^i + \epsilon} \right) \tag{4.4}$$

where the inner bracketed term is the relative load of a node, i.e., akin to Eq. 3.4 (Section 3.2.2). As per the above, all mapping combinations are evaluated (inner loop, Fig. 4.2) to find a feasible mapping with the lowest aggregate load, i.e., $L_{link}(\mathbf{p}_s, \mathbf{q}_s) + L_{node}(\mathbf{v}_t, \mathbf{w}_t)$. Akin to the non-survivable LM scheme (Section 3.2.2) the LM-P approach also avoids congested/costly Layer 2 fog nodes and tries to map more VNFs onto Layer 3 nodes (lower cost). Finally a faster *first fit-LM* (FF-LM-P) scheme is also defined to select the first valid load-based mapping (Figure 4.2).

4.2 Performance Analysis

The survivable SFC mapping heuristic schemes are evaluated using the same multi-layer fog infrastructure detailed in Section 3.5. Namely this scenario has 1,000 terminals (Layer 1), 100 primary fog nodes (Layer 2), and 5 secondary fog nodes (Layer 3). SFC requests are also generated between random terminal nodes, with 4-6 VNFs each, as detailed in Table 3.1. In addition the delay bound (δ_r) is kept at 25 ms to reflect delay-sensitive users, and

tests are run for varying SFC demand batches with infinite request lifetimes. Finally up to K=3 paths are searched in the mapping process (Fig. 4.2).

First, Figure 4.3 plots the blocking rates for both SFC protection strategies. In particular the baseline DM-P and LM-P schemes are selected, along with their first-fit variants, i.e., FF-DM-P and FF-LM-P. As per the findings in Section 3.5.1, the LM scheme still gives the lowest blocking rates (highest carried load) since it spreads out primary and backup VNF mappings across both nodes and layers. By contrast the blocking rate for the DM scheme is much higher, even exceeding the non-exhaustive search first-fit LM protection scheme (FF-LM-P). Furthermore in comparison with results for non-protected demands in Figure 3.4, backup provisioning also gives notably larger separation between the DM and LM strategies. For example the LM-P scheme consistently gives 15-20% less blocking than the DM-P scheme for almost all input loads tested, i.e., versus 5-10% separation between the DM and LM schemes in Figure 3.4.

Average end-to-end delays SFC delays are also shown in Figure 4.4 (averaged across all primary/backup routes for successful SFC demand mappings). Akin to earlier findings in Section 3.5.1, the DM-P scheme gives the lowest delays since it mostly utilizes closer primary fog nodes (at the expense of increased blocking rates). By contrast the LM-P scheme gives notably higher latencies as it selects more dispersed nodes with longer SFC routes, i.e., averaging 6-7 ms higher than the DM-P scheme for most input batch sizes. Additionally the LM-P scheme consistently gives 2-3 ms lower delays than its faster FF-LM-P version, again showing the benefits of more exhaustive mapping search.

Finally average SFC demand embedding cost are also plotted in Figure 4.5, i.e., Eq. 3.10 averaged across primary and backup routes for successful SFC demand mappings. Here the LM-P scheme clearly gives the lowest cost as it uses more secondary fog nodes. As expected the DM-P scheme is much more costly, with per-demand costs averaging almost twice as high as the LM-P scheme. In line with the results for non-protected demands in Figure

3.6, both first-fit strategies (FF-DM-P, FF-LM-P) also have lower costs than the DM-P scheme. Hence many network operators may also prefer load-based provisioning strategies for protected SFC demands in order to maximize their revenues.

Note that the demand sorting schemes presented in Section 3.4 are also evaluated for both the DM-P and LM-P protection heuristics. However the related results are not presented and discussed here since the findings are largely similar, i.e., the LBF and LVF sorting strategies give the lowest demand blocking and end-to-end SFC delays, albeit with higher SFC costs.

Input: Network $\mathbf{G}$=($\mathbf{N}$, $\mathbf{E}$), request $\mathbf{x}_r$=<src_r, dst_r, b_r, δ_r, $\mathbf{F}_r$>
Output: Primary/backup SFC node mappings ($\mathbf{w}^*$, $\mathbf{v}^*$) & Primary/backup SFC routes ($\mathbf{p}^*$, $\mathbf{q}^*$)
/ Find nearest fog nodes for client nodes */*
$src_r^{'} \leftarrow$ closest primary fog node to Layer 1 terminal src_r
$dst_r^{'} \leftarrow$ closest primary fog node to Layer 1 terminal dst_r
/ Prune network to build feasible graph, $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$ */*
$\mathbf{N}' \leftarrow$ Remove nodes $n_j^i \in \mathbf{N}$ with $c_j^i < \min\{f_m \in \mathbf{F}_r\}$ & $\mathbf{E}' \leftarrow$ Remove links $e_{kl}^{ij} \in \boldsymbol{E}$ with $b_{kl}^{ij} < b_r$
Compute K shortest paths from $src_r^{'}$ to $dst_r^{'}$ in $\mathbf{G}'(\mathbf{N}', \mathbf{E}')$, i.e., $\mathbf{P}$=$\{\mathbf{p}_s\}$ s.t. $D_{com}(\mathbf{p}_s) < \delta_r$
Compute K shortest node/link-disjoint paths for each path $\mathbf{p}_s$, i.e., $\mathbf{Q}$=$\{\mathbf{q}_s\}$ s.t. $D_{com}(\mathbf{q}_s) < \delta_r$
/ Order K disjoint path pairs based on scheme */*
if (delay minimization) **then**
    Sort disjoint path pairs {$\mathbf{p}_s$,$\mathbf{q}_s$} by increasing delay, i.e., $D_{com}(\mathbf{p}_s, \mathbf{q}_s) \leq D_{com}(\mathbf{p}_{s+1}, \mathbf{q}_{s+1})$
else if (load balancing) **then**
    Sort disjoint path pairs {$\mathbf{p}_s$,$\mathbf{q}_s$} by increasing load, i.e., $L_{link}(\mathbf{p}_s, \mathbf{q}_s) \leq L_{link}(\mathbf{p}_{s+1}, \mathbf{q}_{s+1})$
/ Initialize path vectors, mapping vectors, and flag */*
$\mathbf{p}^*$={$\emptyset$}, $\mathbf{q}^*$={$\emptyset$}, $\mathbf{v}^*$={<$\emptyset$, $\emptyset$>}, $\mathbf{w}^*$={<$\emptyset$, $\emptyset$>}
$found$=0
/ Search all K disjoint path pairs */*
for (each path pair) **do**
    Extract disjoint path pair {$\mathbf{p}_s$,$\mathbf{q}_s$}
    / Compute max. number of mapping combinations */*
    Q_1=choose($|\mathbf{p}_s|$,$|\mathbf{F}_r|$) */* Primary path */* and Q_2=choose($|\mathbf{q}_s|$,$|\mathbf{F}_r|$) */* Backup path */*
    / Search all combinations */*
    for (each combination) **do**
        / Generate mapping vectors */*
        Compute t-th primary path combination vector, $\mathbf{u}_t$=$\{u_t^m\}$=$\Phi(|\mathbf{p}_s|, |\mathbf{F}_r|, t)$
        Compute t-th backup path combination vector, $\mathbf{y}_t$=$\{y_t^m\}$=$\Phi(|\mathbf{q}_s|, |\mathbf{F}_r|, t)$
        Build t-th primary/backup node mapping vectors ($\mathbf{v}_t$, $\mathbf{w}_t$) from ($\mathbf{u}_t$, $\mathbf{y}_t$)
        / Check delay and resources for mapping */*
        if (($D_{com}(\mathbf{p}_s) + D_{prc}(\mathbf{v}_t) \leq \delta_r$) & ($D_{com}(\mathbf{q}_s) + D_{prc}(\mathbf{w}_t) \leq \delta_r$) &
            ($c_j^i \geq P_m$ for all mapped nodes in $\mathbf{v}_t$, $\mathbf{w}_t$)) **then**
            $found$=1
            / Track & update latest path and mapping vectors */*
            if (first fit-DM or first fist-LB) **then**
                Copy $\mathbf{p}_s \rightarrow \mathbf{p}^*$, $\mathbf{q}_s \rightarrow \mathbf{q}^*$, $\mathbf{v}_t \rightarrow \mathbf{v}^*$, $\mathbf{w}_t \rightarrow \mathbf{w}^*$
                Exit search */* first valid delay/load-based mapping */*
            else if (delay minimization) **then**
                if ($D_{com}(\mathbf{p}_s, \mathbf{q}_s)$+$D_{prc}(\mathbf{v}_t, \mathbf{w}_t)$< $D_{com}(\mathbf{p}^*, \mathbf{q}^*)$+$D_{prc}(\mathbf{v}^*, \mathbf{w}^*)$) **then**
                    Copy $\mathbf{p}_s \rightarrow \mathbf{p}^*$, $\mathbf{q}_s \rightarrow \mathbf{q}^*$, $\mathbf{v}_t \rightarrow \mathbf{v}^*$, $\mathbf{w}_t \rightarrow \mathbf{w}^*$
            else if (load balancing) **then**
                if ($L_{link}(\mathbf{p}_s, \mathbf{q}_s)$+$L_{node}(\mathbf{v}_t, \mathbf{w}_t)$< $L_{link}(\mathbf{p}^*, \mathbf{p}^*)$+$L_{node}(\mathbf{v}^*, \mathbf{w}^*)$) **then**
                    Copy $\mathbf{p}_s \rightarrow \mathbf{p}^*$, $\mathbf{q}_s \rightarrow \mathbf{q}^*$, $\mathbf{v}_t \rightarrow \mathbf{v}^*$, $\mathbf{w}_t \rightarrow \mathbf{w}^*$
/ Reserve resources on paths $\mathbf{p}^*$, $\mathbf{q}^*$ for mappings $\mathbf{w}^*$, $\mathbf{v}^*$ */*
if ($found$) **then**
    Reserve node capacity in $\mathbf{N}$ and link capacity in $\mathbf{E}$

Figure 4.2: Fog-based SFC protection mapping algorithm for a request, $\mathbf{x}_r$

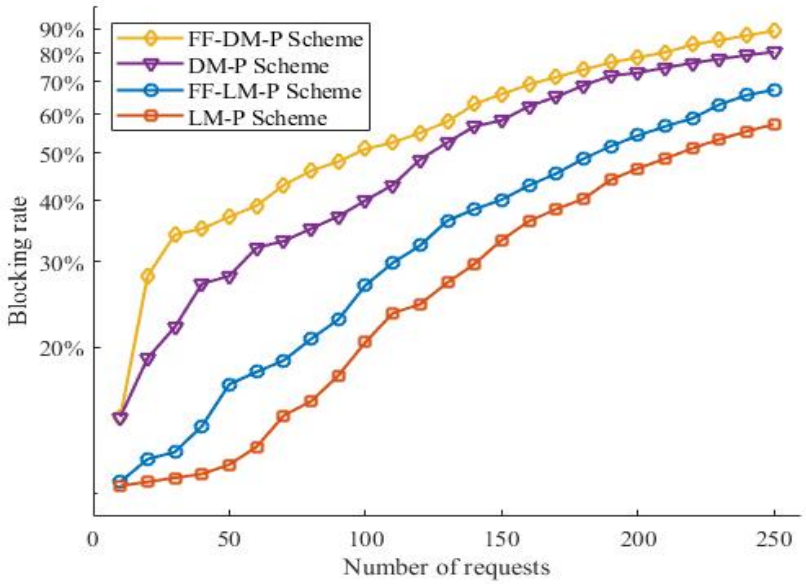

Figure 4.3: Protected SFC demand request blocking rate

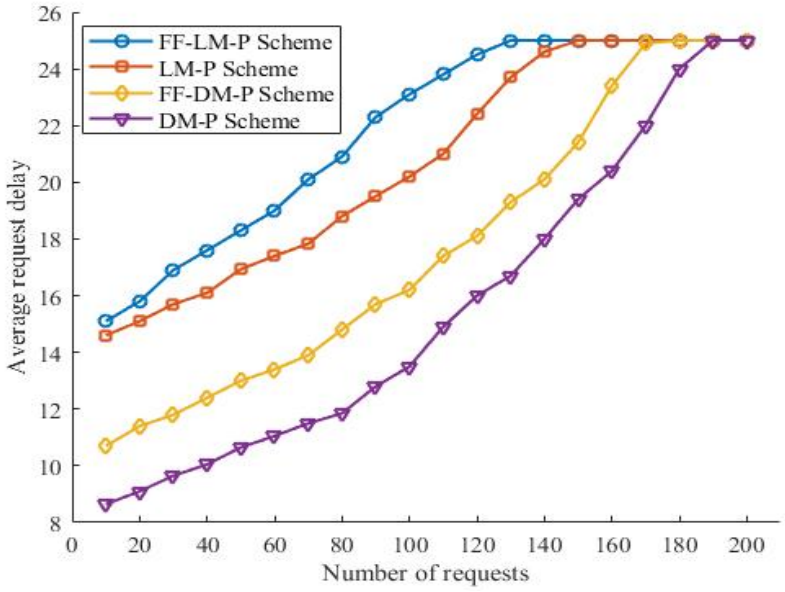

Figure 4.4: Average end-to-end SFC delay (primary, backup)

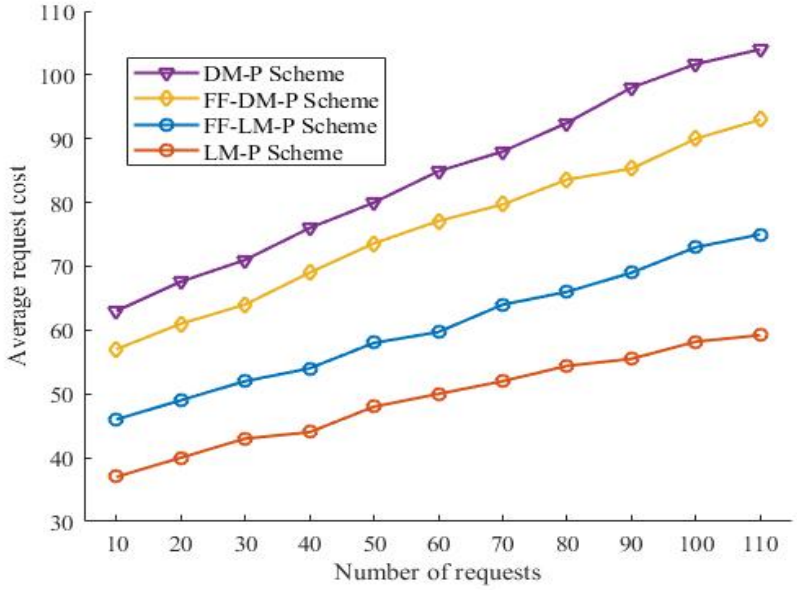

Figure 4.5: Average SFC cost (primary, backup)

Chapter 5: Post-Fault SFC Restoration

Post-fault restoration is classified as a more latent and *non-guaranteed* recovery mechanism, and this approach has been studied extensively to recover failed end-to-end routes in optical and bandwidth connection routing networks [86]. The basic premise of restoration is to detect the occurrence of node and link failures and then attempt dynamic post-fault re-computation of failed demands. In general restoration is much more resource efficient than pre-provisioned protection, i.e., since backup SFC embeddings are not computed and reserved in advance. Additionally, restoration strategies are also more effective against multiple failures, which can easily compromise protection-based schemes designed to handle single failures (Chapter 4). However the performance of any post-fault restoration scheme will be heavily-dependent upon the number of failures, the demand loading of the network infrastructure prior to failures, and also the exact distribution (mapping) of demands and faults across the network.

In light of the above there is a further need to develop post-fault restoration recovery strategies for SFC demands in fog-based networks. These techniques are particularly attractive given the reduced geographic scale of fog networks, i.e., as they can yield faster post-fault signaling (messaging) times and reduced restoration overheads. Hence this chapter introduces some extended notation for post-fault restoration. Subsequently two SFC re-mapping strategies are presented, including intermediate and end-to-end SFC path recovery (leveraging from similar methods studied in bandwidth connection routing networks). Detailed performance analysis is then performed to evaluate these proposed solutions under failure conditions. Carefully note that the actual detection and isolation of node/link faults and

quantification of exact restoration timescales (latencies) is not considered here as these tasks are intricately related to underlying networking protocols and NFV orchestration systems.

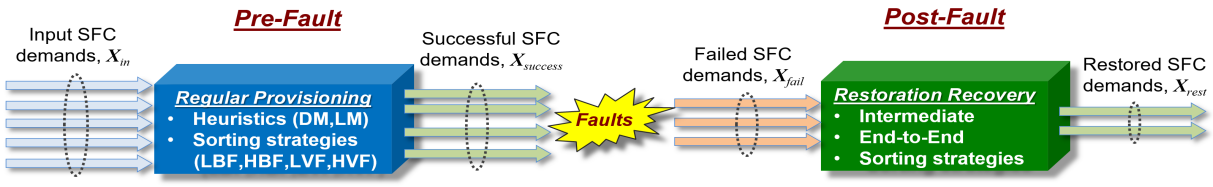

Figure 5.1: Overview of post-fault restoration

5.1 Architecture and Notation Overview

The overall sequence of post-fault restoration is shown in Figure 5.1. Foremost, consider a batch set of incoming SFC demands which is provisioned using any of the regular heuristic schemes detailed in Chapter 3, i.e., to generate a subset of successfully-mapped SFC demands. Upon fault occurrence, a subset of these demands will fail and restoration recovery schemes will be initiated to try to recover them. Now as noted above the exact underlying mechanisms for fault detection/localization are not considered here as they are very implementation specific. Instead, it is assumed that the list of failed infrastructure node/links will be known and provided as input to the restoration phase. Based upon the above, two different post-fault restoration strategies are proposed. Carefully note that this restoration framework only considers single node failures. Specifically this fault type requires both re-mapping of failed VNFs as well as bandwidth connection (or sub-connection) re-routing. However, without loss of generality, this solution can also be adapted to handle multiple node or link failures. In particular the latter fault types are simpler to handle as they may only require SFC connection re-routing.

Consider the requisite notation first. Extending upon the definitions in Section 3.1.2, the failed network node in $\mathbf{G}(\mathbf{N}, \mathbf{E})$ is denoted by $n^{-} \in \mathbf{N}$, and this can either be a primary fog

node (Layer 2) or secondary fog node (Layer 3). Now recall that the input batch of SFC demands is given by the set $\mathbf{X}_{in}$, as per Section 3.1.2. Building upon this, the number of successfully-mapped requests is denoted by $\mathbf{X}_{success}$ (provisioned using any of the heuristic mapping and sorting schemes detailed in Chapter 3). Since a failed node can host multiple VNFs, the subset of successful SFC mappings affected by the failure is denoted by the set of failed SFC demands, $\mathbf{X}_{fail} \subseteq \mathbf{X}_{success}$. Furthermore the specific VNF node mapping that fails within a demand $\mathbf{x}_r \subset \mathbf{X}_{fail}$ is also denoted by f_r^-. Finally, the set of successfully-restored SFC demands is given by $\mathbf{X}_{rest} \subseteq \mathbf{X}_{fail}$. These set entities are also shown in Figure 5.1.

5.2 Failed SFC Re-Mapping

The overall psueducode for post-fault restoration is presented in Figure 5.2. Specifically, the algorithm starts by (optionally) sorting the incoming set of failed SFC demands, $\mathbf{X}_{fail}$, based upon any of the schemes outlined in Section 3.4 (HBF, LBF, HVF, or LVF). This step yields an ordered set of failed SFC demands, denoted by $\mathbf{X}'_{fail}$. Next a working graph, $\mathbf{G}''(\mathbf{N}'', \mathbf{E}'')$, is constructed by taking $\mathbf{G}(\mathbf{N}, \mathbf{E})$ and removing the failed node n^- and all of its links. In addition for the case of end-to-end restoration, all node and link capacity reservations pertaining to *non-failed* nodes and links for failed SFC demands (in $\mathbf{X}'_{fail}$) are also relinquished, i.e., by appropriately incrementing free capacity levels. The restoration algorithm then loops through all failed SFC demands in $\mathbf{X}'_{fail}$ and tries to recover each one using an *end-to-end* or *intermediate* restoration strategy (detailed subsequently).

Now the actual re-mapping procedure makes full re-use of the heuristic SFC embedding algorithms defined earlier in Chapter 3, i.e., DM or LM provisioning. Namely a temporary "input" SFC request, $\mathbf{x}'$, is defined as per the desired recovery mapping strategy, and an appropriate call made to the algorithm in Figure 3.3. Finally all failed SFC demands in $\mathbf{X}'_{fail}$ that are successfully re-mapped (recovered) are then moved to the set of restored SFC demands, $\mathbf{X}_{success}$. Carefully note that the heuristic approach in Figure 3.3 already reserves

Input: Network $\mathbf{G}$=($\mathbf{N}, \mathbf{E}$), failed node n^{-}, failed SFC demand set $\mathbf{X}_{fail}$
Output: Restored SFC set $\mathbf{X}_{rest}$
/ Sort failed SFC demand set (optional) */*
$\mathbf{X}^{'}_{fail} \leftarrow$ Sort demands in $\mathbf{X}_{fail}$ (as per strategies in Section 3.4)
/ Build working topology by pruning failed node and its links from network */*
$\mathbf{N}^{''} \leftarrow$ Remove node n^{-}
$\mathbf{E}^{''} \leftarrow$ Remove all links in $\mathbf{E}$ connected to n^{-}
/ For end-to-end restoration also free all resources used by failed demands */*
if (end-to-end restoration) **then**
  for (each failed SFC demand) **do**
    $\mathbf{N}^{''} \leftarrow$ Free all resources in $\mathbf{N}$ used by *non-failed* nodes in $\mathbf{x}_r \in \mathbf{X}_{fail}$
    $\mathbf{E}^{''} \leftarrow$ Free all resources in $\mathbf{E}$ used by *non-failed* links in $\mathbf{x}_r \in \mathbf{X}_{fail}$
/ Initialized restored SFC demand set */*
$\mathbf{X}_{rest}$={$\emptyset$}
/ Process and attempt restoration for all failed demands */*
for (each failed SFC demand) **do**
  Extract demand $\mathbf{x}_r \in \mathbf{X}^{'}{}_{fail}$
  / Define modified SFC demand tuple */*
  if (end-to-end restoration) **then**
    / All VNFs between ingress/egress primary fog nodes */*
    $\mathbf{x}^{'} = \mathbf{x}_r = <src_r, dst_r, b_r, \delta_r, \mathbf{F}_r>$
  else if (intermediate restoration) **then**
    / Single VNF between adjacent nodes to fault */*
    $src^{'}_r \longleftarrow$ Anterior node to n^{-}
    $dst^{'}_r \longleftarrow$ Posterior node to n^{-}
    $\delta^{'}_r = \delta_r -$ delays on working segments */* Reduced delay bound for segment */*
    $\mathbf{F}^{'}_r = \{f^{-}_r\}$ */* Failed VNF node mapping */*
    $\mathbf{x}^{'} = <src^{'}_r, dst^{'}_r, b_r, \delta^{'}_r, \mathbf{F}^{'}_r>$
  Run SFC embedding routine for request $\mathbf{x}^{'}$ (Figure 3.3) over $\mathbf{G}^{''}(\mathbf{N}^{''}, \mathbf{E}^{''})$
  / Update restored set */*
  if (new mapping found for $\mathbf{x}^{'}$) **then**
    Move $\mathbf{x}_r \longrightarrow \mathbf{X}_{rest}$ */* Note resources already updated in* $\mathbf{G}^{''}(\mathbf{N}^{''}, \mathbf{E}^{''})$ **/*

Figure 5.2: Fog-based SFC restoration for failed demands (end-to-end, intermediate)

resources for all successfully-provisioned demands, and hence free node and link capacities in $\mathbf{N}''$ and $\mathbf{E}''$ do not need to be decremented further upon return. The two different restoration strategies are now presented.

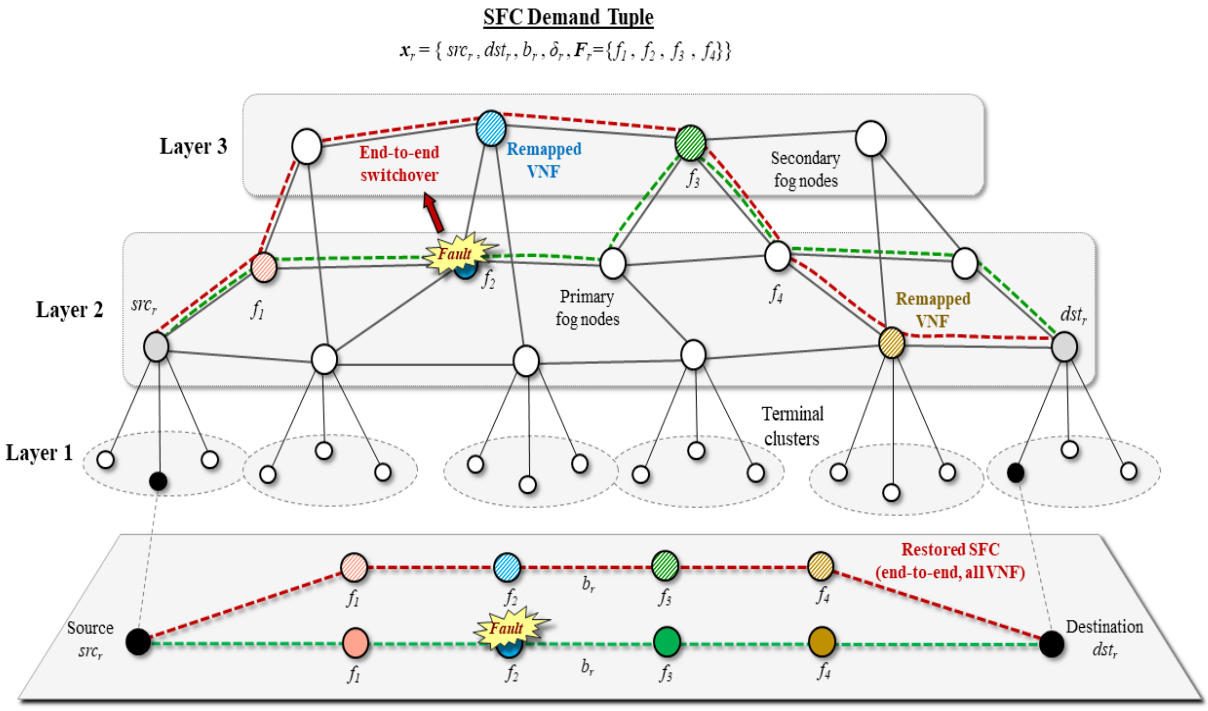

Figure 5.3: End-to-end SFC demand restoration example

5.2.1 End-to-End SFC Restoration

End-to-end restoration is a straightforward approach that simply tries to re-establish the complete failed SFC request. Hence the heuristic SFC embedding algorithm (Figure 3.3) is called with a temporary "input" SFC request equal to the exact failed demand, i.e., $\mathbf{x}' = \mathbf{x}_r \subseteq \mathbf{X}_{failed}$. Expectedly this strategy is more computationally-intensive since it must completely re-map all VNFs and their interconnecting bandwidth connection segments, i.e.,

even those which are unaffected by the failed node n^-. However, this "clean" re-mapping approach can also give increased resource efficiency, particularly link bandwidth.

An example of end-to-end SFC restoration is shown in Figure 5.3 for the same 4-VNF SFC embedding example presented earlier in Figure 3.2. Here it is assumed that the primary fog node hosting the second VNF, f_2, in Layer 2 fails. Hence the complete SFC is re-mapped, and this can result in some of the same VNF placements as per the original SFC mapping. For example VNFs f_1 and f_3 in Figure 5.3 are still re-mapped to their original locations, whereas VNFs f_2 and f_4 are now placed at new locations.

5.2.2 Intermediate SFC Restoration

Intermediate restoration is a more expedient approach which only tries to recover the failed VNF, i.e., by keeping in place all existing non-failed VNF mappings in a failed SFC demand. To achieve this the heuristic embedding algorithm in Figure 3.3 is called using a temporary "input" SFC request containing the singled failed VNF. Namely the appropriate source and destination end-points for this SFC request are specified as the (non-failed) anterior and posterior embedding nodes to the failed node n^-, denoted by src' and dst', respectively. These end-point nodes are used to "stitch" and re-connect the restored VNF node mapping with the existing non-failed SFC segments. Carefully note that the delay bound for this modified single-VNF SFC demand is also reduced appropriately from δ_r in order to account for the delay budgets of existing nodes and links (on the non-failed portions of the embedded SFC).

An example of intermediate SFC restoration is also shown in Figure 5.4 for the same 4-VNF SFC embedding example presented earlier in Figure 3.2. Namely the primary node hosting the second VNF, f_2, in Layer 2 fails and this causes a re-mapping of the SFC segment between the anterior node (mapping VNF f_1) and posterior node (mapping VNF f_3). In particular the restored VNF is now placed in the secondary fog layer, and the intermediate

sub-connection route (reconnecting to the remaining working VNFs in the SFC embedding) is 3 hops long.

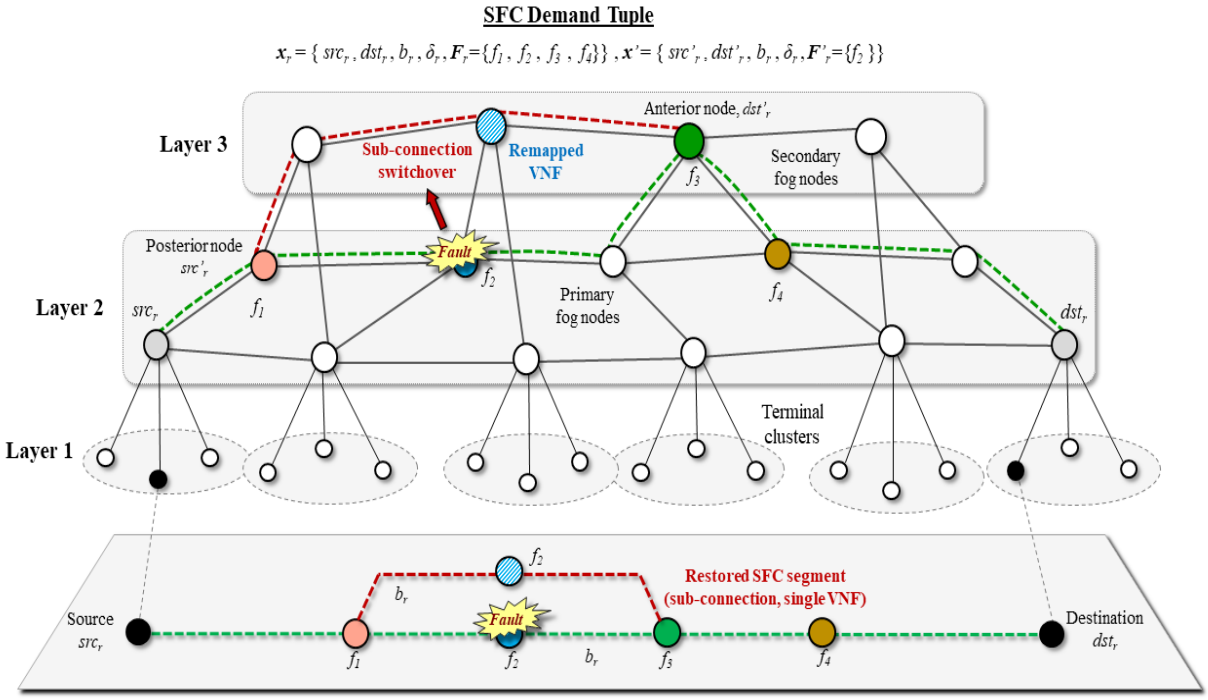

Figure 5.4: Intermediate SFC demand restoration example

5.3 Performance Analysis

Post-fault restoration recovery is tested using the same multi-layer topology from Chapters 3 and 4. As per the network topology defined in Section 3.5, there are 100 primary fog nodes (Layer 2) and 5 secondary fog nodes each handling a "cluster" of 20 primary fog nodes (Layer 3). Similarly, a total of T=6 VNFs are defined, and each SFC request has 4-6 VNFs with 4-6 bandwidth units demand (b_r) and a delay bound (δ_r) of 25 ms (see Table 3.1). The number of computed shortest-paths is also set to K=3 (Figure 3.3). Furthermore, tests are done for single isolated node faults at both Layer 2 (primary fog nodes) and Layer 3

(larger secondary fog nodes carrying more demands). Finally SFC re-mapping computation is tested for both the baseline DM and LM schemes (Section 3.2) as well as their LBF sorting variants for failed demands in $\mathbf{X}'_{fail}$ (Section 3.4.2). This particular variant is chosen since it has been shown to give increased SFC setup success rates, as per the results in Section 3.5.2. The detailed findings are now presented.

Foremost, Figures 5.5 and 5.6 summarize the number of post-fault failures for both the DM and LM schemes. Overall these plots represent the absolute numbers of successfully-provisioned SFC requests that fail after a node fault at Layer 2 or Layer 3, i.e., $|\mathbf{X}_{fail}|$, Figure 5.1. As expected these results confirm that the number of failed demands increases with input load (measured by the number of successfully-routed SFC demands in the network prior to failure, $|\mathbf{X}_{success}|$). Furthermore these results also indicate that Layer 3 nodes failures are less impactful than smaller Layer 2 node failures at lower loads. However at increased load regimes, Layer 3 node failures yield much higher SFC demand failures. This inflexion/crossover is expected as Layer 3 nodes carry more (less) demands at increased (reduced) loads. Furthermore the onset of this inflexion occurs earlier with the LM scheme at it tends to be more reliant on secondary nodes. By contrast, the DM scheme is less susceptible to secondary node failures, i.e., peak of 25 failures with the DM-LBF scheme versus almost 40 failures with the LM-LBF scheme. In fact DM mappings do not experience any failures at very light input loads since they do not use any Layer 3 nodes at these load regimes, e.g,. for under 20 input requests, Figure 5.5. Finally these results also indicate a small but consistent increase in the number of failures with the LBF-based schemes. Again this increase is due to the fact that these variants embed more SFC demands than their baseline counterparts.

Next, the post-fault restoration schemes are gauged to evaluate their effectiveness in recovering failed SFC demands. Although runs are done with both the baseline (DM, LM) schemes and their LBF variants from Chapter 3, only findings with the latter variants are presented in the remainder of this chapter as the respective trends are similar. Accordingly

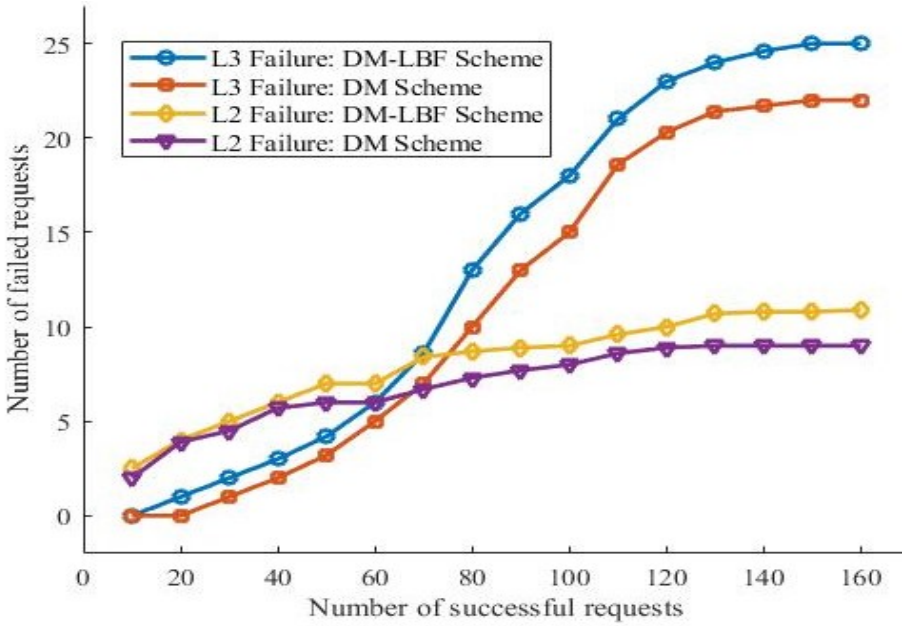

Figure 5.5: Number of failed requests for DM scheme

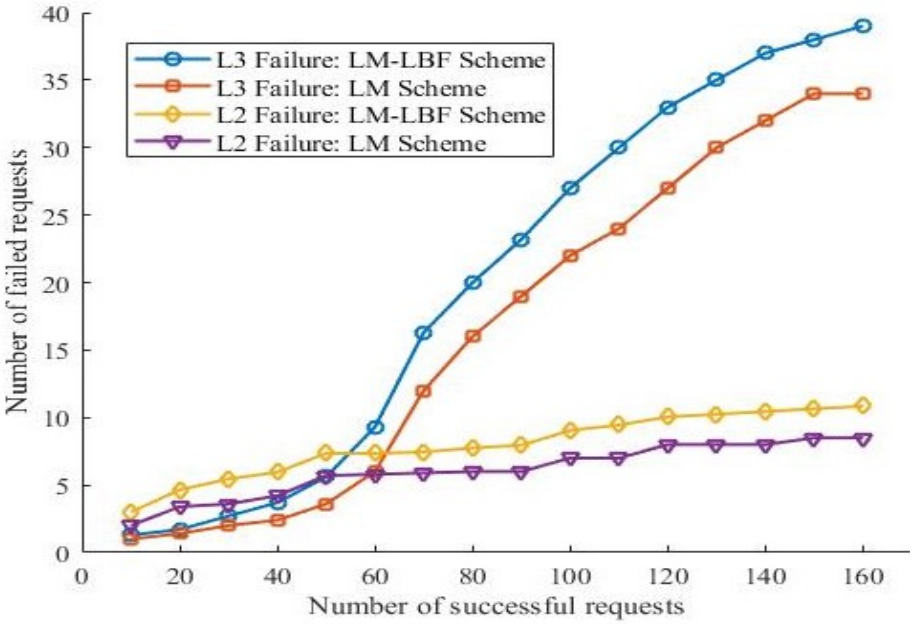

Figure 5.6: Number of failed requests for LM scheme

Figures 5.7 and 5.8 plot the restoration rate percentages for both intermediate and end-to-end restoration. These values represent the proportion of failed requests (from the subset $\mathbf{X}_{fail}$) that are successfully restored after restoration, i.e., $|\mathbf{X}_{success}|$, Figure 5.1. Overall

these findings reveal some key findings. Foremost, end-to-end restoration consistently outperforms intermediate restoration for both Layer 2 and Layer 3 node failures. Specifically the improvement in restoration success rates averages 5-10% across all input load regimes, measured by the number of failed connections. This improvement is due to the fact that end-to-end restoration is better at distributing SFC routes across the entire network as opposed to the immediate vicinity of the failed node. In general this finding mirrors earlier results for bandwidth routing networks where end-to-end connection recovery also gives improved success rates versus intermediate restoration, see [86]. These results also show that recovery success rates decline with increasing load, i.e., as there is more resource contention due to increased numbers of failed SFC demands. However these declines are much more drastic for larger secondary node (Layer 3) faults for both restoration strategies. Namely restoration rates for Layer 3 faults are very high at lighter load regimes, averaging well over 80% for both the DM and LM-based embedding strategies. However, the respective recovery rates decline to the 50% range at higher load regimes. By contrast, post-fault restoration for Layer 2 faults is slightly more effective at higher loads, i.e., restoration rates averaging 12-15% higher with the DM-LBF (LM-LBF) schemes. This slight increase is due to the reduced number of (failed) demands carried by smaller primary fog nodes.

Restoration recovery re-computes failed VNF embeddings and generally gives altered SFC routes. Hence it is also important to measure the impact on end-to-end delay. Accordingly, Figures 5.9 (DM-LBF scheme) and 5.10 (LM-LBF scheme) plot the average *increase* in end-to-end SFC delays versus input loads (for both Layer 2 and Layer 3 node failures). Overall the results for both the DM-LBF and LM-LBF schemes give very similar findings here. Foremost all restoration schemes show a clear increase in SFC delay under heavier loads, i.e., as increased link congestion forces longer intermediate or end-to-end SFC routes. In addition secondary fog node (Layer 3) faults are much more impactful, with average delays rising by about 6-7 ms for both restoration strategies at high loads versus only 3-4 ms for

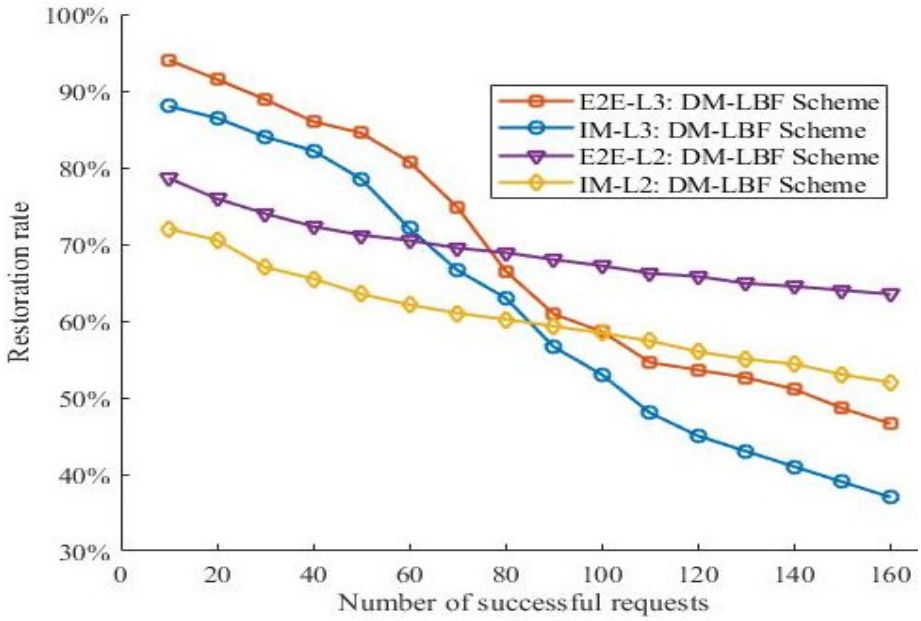

Figure 5.7: Restoration rate for DM-LBF scheme

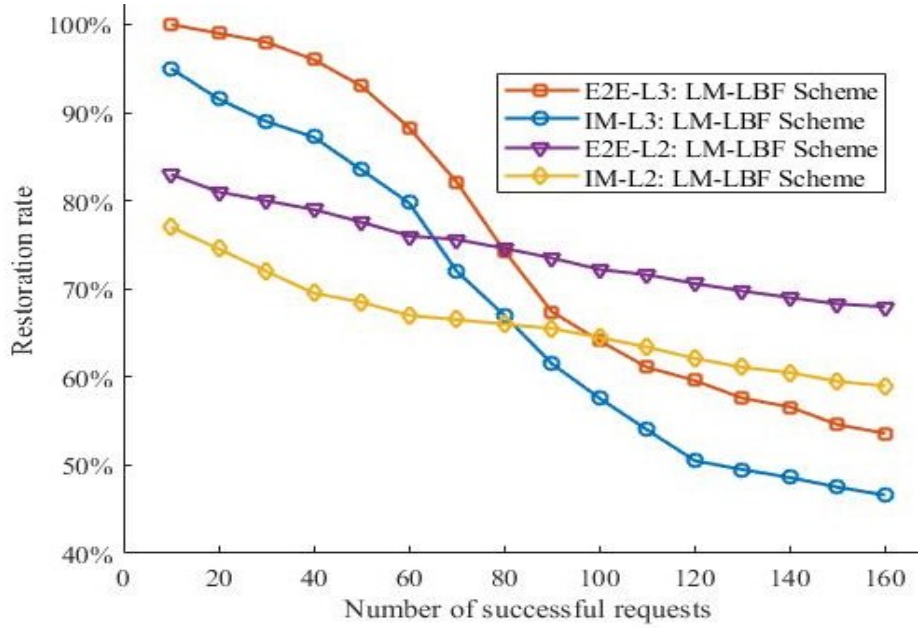

Figure 5.8: Restoration rate for LM-LBF scheme

Layer 2 node faults. These increases represent approximately 25% and 15% of the 25 ms SFC delay bound value, respectively. However at very light loads, Layer 3 node faults yield slightly smaller delay increases (in 0.5-1 ms range), i.e., since they can be recovered using

closer primary fog nodes. Furthermore end-to-end restoration consistently gives smaller delay increases than intermediate restoration, i.e., averaging about 0.5 ms for both the LM-LBF and DM-LBF strategies. Again these findings mirror results from earlier connection restoration studies where end-to-end restoration gives notably lower delays (hop counts) than intermediate restoration, see [86].

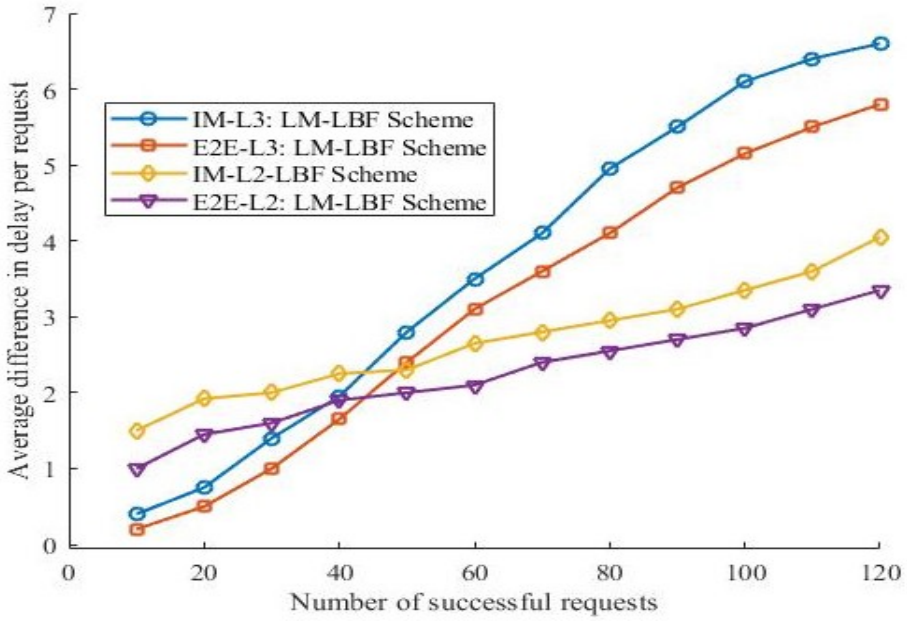

Figure 5.9: Average increase in delay of restored demands (DM-LBF scheme)

Finally, Figures 5.11 and 5.12 plot the average cost increase for recovered SFC demands for both restoration strategies. Akin to the results for delay increase, both the DM-LBF and LM-LBF schemes give very similar performances for Layer 2 and Layer 3 node faults. Foremost all restoration schemes give higher post-fault SFC costs, particularly at higher input loads. For example the average cost increase for Layer 2 node faults at peak load is about 4-5 units and about 9-11 units for Layer 3 node faults. These increases represent approximately 10% more cost after failure, i.e., as determined by comparing the cost increases to respective cost ranges in Figures 3.11 and 3.12. In addition, end-to-end restoration is slightly more cost-effective than intermediate restoration since is uses fewer link resources.

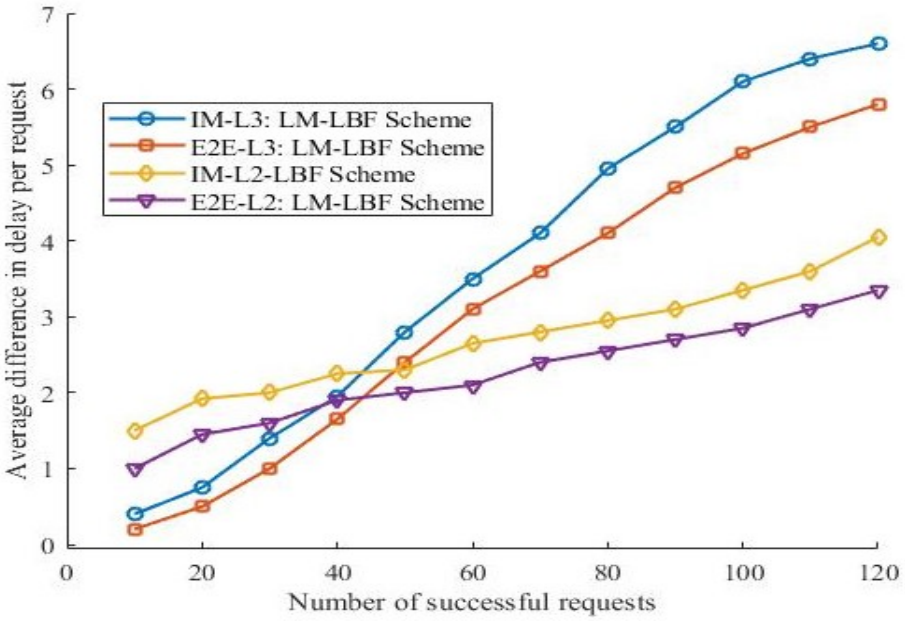

Figure 5.10: Average increase in delay of restored demands (LM-LBF scheme)

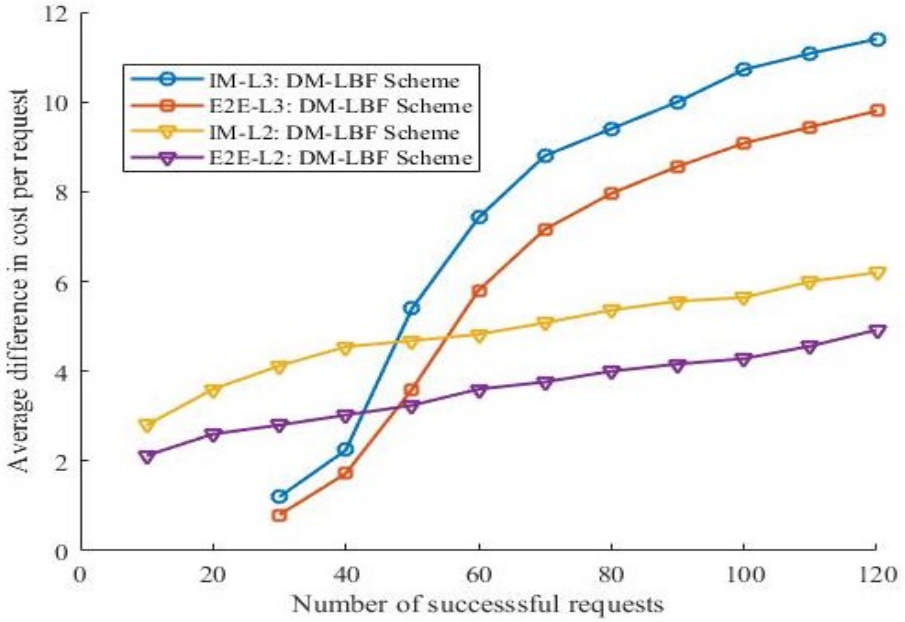

Figure 5.11: Average increase in cost of restored demands (DM-LBF scheme)

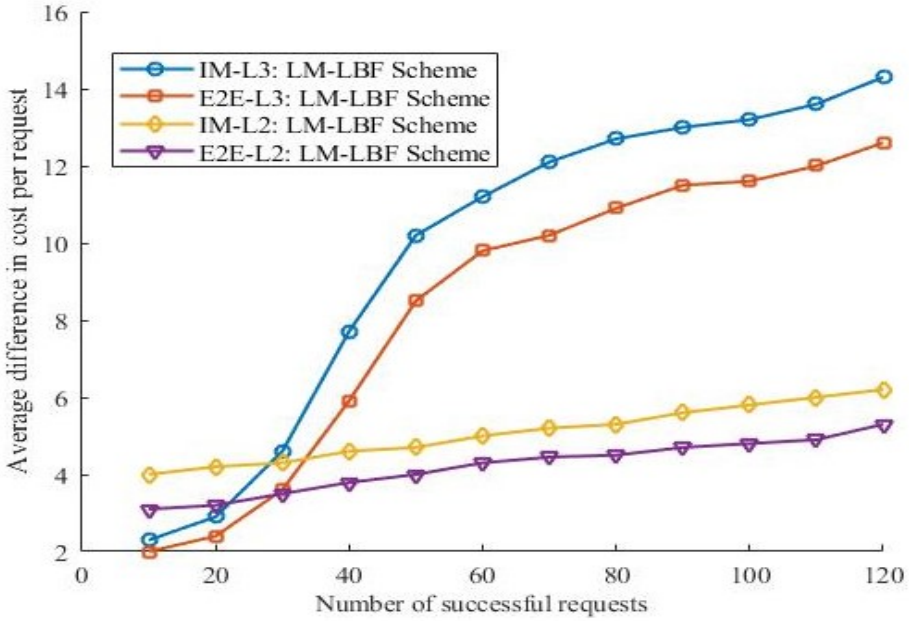

Figure 5.12: Average increase in cost of restored demands (LM-LBF scheme)

Chapter 6: Conclusions

This dissertation addresses the topic of SFC demand provisioning in specialized multi-layer fog networks. First, Chapter 2 presents a survey of related work on SFC provisioning and survivability, with a focus on fog-based setups. Next, Chapter 3 presents some novel polynomial-time heuristic methods for provisioning SFC demands in realistic two-layer fog networks. Extending upon this Chapter 4 introduces new pre-provisioned protection algorithms to achieve survivability against single fault scenarios. Finally Chapter 5 details novel post-fault restoration methods to further improve survivability support. The key findings from this research effort are now summarized along with potential future research directions.

6.1 Summary of Research Findings

This dissertation introduces a range of polynomial-time algorithms to achieve varying provisioning objectives for SFC demands in fog computing networks. Commensurate fault recovery methods are also proposed in order to improve service survivability. A summary of the major findings of this dissertation work are now presented.

The findings in Chapter 3 presents some novel on-line heuristic algorithms to map SFC demands in two-layer fog infrastructures. These schemes utilize k-shortest path routing algorithms to compute a set of candidate SFC routes and then search them for multiple VNF mapping combinations. In particular two SFC mapping strategies are developed here based upon delay and load minimization. The major findings from this effort are as follows:

- Delay-based mapping scheme (DM) always gives the lowest end-to-end latency for all scenarios tested. However this improvement comes at the expense of significantly higher blocking rates and average SFC costs, i.e., as this scheme tends to overload Layer 2 nodes/links with smaller resource pools and higher costs.

- Load minimization mapping scheme (LM) gives much lower blocking rates than the delay minimization scheme as it maps demands across multiple nodes in both Layers 2 and 3. Although the average delay performance is worse than the DM scheme, this method also yields notably lower SFC resource costs, i.e., almost half of the DM scheme. As a result the LM scheme may be more favorable for many network operators.

- Least resource first sorting strategies that prioritize smaller bandwidth or node resource demands (such as LBF and LVF) yield much lower blocking rates than counterpart strategies which prioritize larger demands first (such as HBF and HVF). The average end-to-end delays are also lower with these sorting strategies.

- Least resource first sorting strategies also yield higher average SFC costs, i.e., since these schemes map increased numbers of smaller demands onto more costly primary layer nodes and links.

Next, Chapter 4 addresses the critical problem of SFC survivability and studies pre-fault protection strategies. In particular the focus here is on single node faults which are by far the most common failure occurrences. Accordingly the delay and load minimization heuristic strategies in Chapter 3 are further augmented to compute link/node-disjoint backup SFC embeddings. The overall results here indicate:

- Load minimization protection scheme (LM-P) gives less blocking rates, i.e., higher provisioning success, for equivalent input loads compared to the delay minimization

protection scheme (DM-P). In fact this reduction is much larger than that observed for non-protected SFC demands between the DM and LM strategies.

- Delay minimization protection scheme (DM-P) also gives the lowest end-to-end SFC delays as it tends to map fewer primary/backup SFC routes onto closer Layer 2 nodes. However the associated costs are also notably higher, i.e., almost double, akin to the increase between the LM and DM schemes for non-protected SFC demands.

Finally, Chapter 5 extends upon the topic of SFC survivability and studies more flexible post-fault restoration methodologies. In particular these schemes are more resource efficient than pre-fault protection and can also handle multiple faults. Hence two different algorithms are presented to recover failed SFC demands in fog setups, i.e., end-to-end and intermediate SFC recovery. The overall findings indicate:

- Post-fault SFC restoration gives very good recovery success rates in multi-layer fog networks, i.e., ranging anywhere from 70-95% at lighter loads to 40-70% at heavier loads. Hence this approach provides a very viable survivability solution for network operators.

- End-to-end SFC restoration is more effective than intermediate restoration across all input load regimes evaluated, i.e., averaging 5-10% higher restoration success rates. In addition the corresponding increases in average end-to-end SFC delay and cost are also lower with this approach.

- Secondary fog node faults (Layer 3) are much more impactful under increased load conditions for both the DM and LM provisioning strategies. Layer 3 faults also yield lower restoration success rates (and increased costs) under higher load conditions. This reduction (increase) is due to the increased level of demand multiplexing at the nodes.

6.2 Extensions and Future Directions

In conclusion this dissertation research provides a strong foundation for provisioning NFV-based services in fog computing networks. The proposed methods are practical and scalable from a run-time perspective, and thereby lend well to real-world application. Based upon these contributions, a range of further and more expansive topic areas can also be investigated. Foremost more formalized optimization models can be developed for batch provisioning and protection of SFC requests in fog networks. Although various studies have proposed such methods for SFC mapping in cloud networks, the specialized infrastructure requirements and resource limitations of fog domains need careful consideration. In addition it is important to note that such optimization-based methods may only be feasible for very small infrastructures and demand pools, i.e., due to high variable count intractability. As a result further metaheuristic schemes (such as those based upon simulated annealing and genetic algorithms) can also be investigated to provide a median approach between optimization and adhoc heuristic strategies.

Finally, with regards to service survivability, more efficient shared protection schemes can also be considered in order to improve the resource efficiency (and lower the cost) of SFC demands. Further investigations can also integrate the pre-fault protection and post-fault restoration mechanisms introduced in this thesis to build more robust capabilities. Specifically post-fault restoration can be used to overcome the rare case of multiple faults and further augment the survivability performance of single-fault protection schemes.

References

[1] C. Rotsos *et al.* Network Service Orchestration Standardization: A Technology Survey. *Computer Standards & Interfaces*, 54:203–215, 2017.

[2] E. Hernandez-Valencia, S. Izzo, and B. Polonsky. How Will NFV Transform Service Provider Opex? *IEEE Network*, 29(3):60–67, 2015.

[3] R. Majumbi *et al.* Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys and Tutorials*, 18(1):236–262, 2016.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. Association for Computing Machinery, 2012.

[5] M. Abdelshkour. IoT from Cloud to Fog Computing. In *Cisco Blogs: Perspectives*, 2015.

[6] Tom Hao Luan, Longxiang Gao, Zhi Li, Yang Xiang, and Limin Sun. Fog Computing: Focusing on Mobile Users at the Edge. *ArXiv*, abs/1502.01815, 2015.

[7] D. Zhao, D. Liao, G. Sun, and S. Xu. Towards Resource-Efficient Service Function Chain Deployment in Cloud-Fog Computing. *IEEE Access*, 6:66754–66766, 2018.

[8] A. Taherkordi and F. Eliassen. Poster Abstract: Data-centric IoT Services Provisioning in Fog-Cloud Computing Systems. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 317–318. ACM, 2017.

[9] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. A Survey on Service Function Chaining. *Network and Computer Applications*, 75(c):138–155, 2016.

[10] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo. Towards Making Network Function Virtualization a Cloud Computing Service. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 89–97, 2015.

[11] E. Amaldi, S. Coniglio, A. Koster, and M. Tieves. Network Slicing for Service-Oriented Networks Under Resource Constraints. *Electronic Notes in Discrete Mathematics*, 52:213–220, 2016.

[12] 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*, 18(1):236–262, 2016.

[13] J. Gil Herrera and J. F. Botero. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[14] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal. NFV: State of the Art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC). *IEEE Network*, 28(6):18–26, 2014.

[15] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network Function Virtualization: Challenges and Opportunities for Innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[16] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari. Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining. *IEEE Transactions on Network and Service Management*, 16(1):374–388, 2019.

[17] D. bhamare *et al.* Optimal Virtual Network Function Placement in Multi-Cloud Service Function Chaining Architecture. *Computer Communication*, 102:1–16, 2017.

[18] N. Ghani D. Oliveira, J. Crichigno. On Sensitive and Weighted Routing and Placement Schemes for Network Function Virtualization. *Infocommunications Journal*, 9(4):15–23, 2017.

[19] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, and T. Wood. Virtual Function Placement and Traffic Steering in Flexible and Dynamic Software Defined Networks. In *Structural Information and Communication Complexity*, pages 104–118. IEEE, 2015.

[20] T. Lukovszki and S. Schmid. Online Admission Control and Embedding of Service Chains. In *IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, pages 104–118, 2014.

[21] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and S. Davy. Design and Evaluation of Algorithms for Mapping and Scheduling of Virtual Network Functions. In *2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9. IEEE, 2015.

[22] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau. Online VNF Scaling in Datacenters. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 140–147. IEEE, 2016.

[23] D. Cho, J. Taheri, A. Y. Zomaya, and L. Wang. Virtual Network Function Placement: Towards Minimizing Network Latency and Lead Time. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2017.

[24] D. Bhamare, R. Jain, M. Samaka, G. Vaszkun, and A. Erbad. Multi-Cloud Distribution of Virtual Functions and Dynamic Service Deployment: Open ADN Perspective. In *2015 IEEE International Conference on Cloud Engineering*. IEEE, 2015.

[25] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca. The Dynamic Placement of Virtual Network Functions. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014.

[26] 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 *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017.

[27] N Siasi, M Jasim, J Crichigno, and N Ghani. Container-based Service Function Chain Design and Mapping. In *IEEE SoutheastCon 2019*, pages 1–6. IEEE, 2019.

[28] M. Gharbaoui, C. Contoli, G. Davoli, G. Cuffaro, B. Martini, F. Paganelli, W. Cerroni, P. Cappanera, and P. Castoldi. Demonstration of Latency-Aware and Self-Adaptive Service Chaining in 5G/SDN/NFV Infrastructures. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–2, 2018.

[29] N. Zhang, Y. Liu, H. Farmanbar, T. Chang, M. Hong, and Z. Luo. Network Slicing for Service-Oriented Networks Under Resource Constraints. *IEEE Journal on Selected Areas in Communications*, 35(11):2512–2521, 2017.

[30] Ahmet Cihat Baktir, Atay Ozgovde, and Cem Ersoy. How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases, and Future Directions. *IEEE Communications Surveys & Tutorials*, 19:2359–2391, 2017.

[31] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: Bringing the Cloud to the Mobile User. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, New York, NY, USA, 2012. Association for Computing Machinery.

[32] N. M. Gonzalez *et al.* Fog Computing: Data Analytics and Cloud Distributed Processing on the Network Edges. In *35th International Conference of the Chilean Computer Science Society (SCCC)*, 2016.

[33] Pengfei Hu, Sahraoui Dhelimabc, Huansheng Ningabc, and Tie Qiud. Survey on Fog Computing: Architecture, key Technologies, Applications and Open Issues. *IET Networks*, 98:27–42, 2017.

[34] Jordan Shropshire. Extending the Cloud with Fog: Security Challenges & Opportunities. In *2014 Americas Conference on Information Systems (AMCIS)*, 2014.

[35] Mithun Mukherjee, Lei Shu, and Di Wang. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. *IEEE Communications Surveys & Tutorials*, 20(3):1826–1857, 2018.

[36] N. Siasi, A. Jaesim, and N.Ghani. Deep Learning for Service Function Chain Provisioning in Fog Computing. In *IEEE Access, Vol. 99, No.11*, pages x–x.

[37] M. Aazam and E. Huh. Fog Computing: The Cloud-IoTIoE Middleware Paradigm. *IEEE Potentials*, 35(3):40–44, 2016.

[38] Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities. In *2015 Proceedings of the ASE Big Data & Social Informatics*, New York, NY, USA, 2015. Association for Computing Machinery.

[39] S. Sarkar and S. Misra. Theoretical Modelling of Fog Computing: a Green Computing Paradigm to Support IoT Applications. *IET Networks*, 5(2):23–29, 2016.

[40] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[41] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen. Help Your Mobile Applications with Fog Computing. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)*. IEEE, 2015.

[42] S. He, B. Cheng, H. Wang, X. Xiao, Y. Cao, and J. Chen. Data Security Storage Model for Fog Computing in Large-Scale IoT Application. In *2018 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018.

[43] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. Ben Hadj-Alouane, M. J. Morrow, and P. A. Polakos. A platform as-a-service for hybrid cloud/fog environments. In *IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2016.

[44] O. Bibani, C. Mouradian, S. Yangui, R. H. Glitho, W. Gaaloul, N. B. Hadj-Alouane, M. Morrow, and P. Polakos. A Demo of IoT Healthcare Application Provisioning in Hybrid Cloud/Fog Environment. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016.

[45] Nardelli-M. Schulte S. et al. Skarlat, O. Optimized IoT Service Placement in the Fog. *Service Oriented Computing and Applications*, 11(4):427–443, 2017.

[46] A. Yousefpour. QoS-Aware Dynamic Fog Service Provisioning. *ArXiv*, abs/1802.00800, 2018.

[47] Carla Mouradian, Somayeh Kianpisheh, and Roch Glitho. Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 25–30. IEEE, 2018.

[48] Eduardo S Gama, Roger Immich, and Luiz F Bittencourt. Towards a Multi-Tier Fog/Cloud Architecture for Video Streaming. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 13–14. IEEE, 2018.

[49] B. Donassolo *et al.* Fog Based Framework for IoT Service Provisioning. *IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019.

[50] N Siasi, A Jaesim, and N Ghani. Service Function Chain Provisioning Schemes for Multi-Layer Fog Networks. *IEEE Networking Letters*, 2(1):38–42, 2020.

[51] N. Siasi, A. Jaesim, and N. Ghani. Tabu Search for Efficient Service Function Chain Provisioning in Fog Networks. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 145–150, 2019.

[52] Nguyen B Truong, Gyu Myoung Lee, and Yacine Ghamri-Doudane. Software Defined Networking-based Vehicular Adhoc Network with Fog Computing. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1202–1207. IEEE, 2015.

[53] R. Chaudhary, N. Kumar, and S. Zeadally. Network Service Chaining in Fog and Cloud Computing for the 5G Environment: Data Management and Security Challenges. *IEEE Communications Magazine*, 55(11):114–122, 2017.

[54] Denis do Rosário, Matias Artur Klafke Schimuneck, João Camargo, Jéferson Campos Nobre, Cristiano Both, Juergen Rochol, and Mario Gerla. Service Migration from Cloud to Multi-tier Fog Nodes for Multimedia Dissemination with QoE Support. *Sensors*, 18(2), 2018.

[55] Salma Matoussi, Ilhem Fajjari, Salvatore Costanzo, Nadjib Aitsaadi, and Rami Langar. A User Centric Virtual Network Function Orchestration for Agile 5G Cloud-RAN. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018.

[56] Yu Qiu, Haijun Zhang, Keping Long, Hongjian Sun, Xuebin Li, and Victor CM Leung. Improving Handover of 5G Networks by Network Function Virtualization and Fog computing. In *2017 IEEE International Conference on Communications in China (ICCC), Qingdao, China*, pages 1–5. IEEE, 2017.

[57] Jingchu Liu, Sheng Zhou, Jie Gong, Zhisheng Niu, and Shugong Xu. Graph-Based Framework for Flexible Baseband Function Splitting and Placement in C-RAN. In *2015 IEEE International Conference on Communications (ICC), London, UK*, pages 1958–1963. IEEE, 2015.

[58] I. B. B. Harter, D. A. Schupke, M. Hoffmann, and G. Carle. Network virtualization for Disaster Resilience of Cloud Services. *IEEE Communications Magazine*, 52(12):88–95, 2014.

[59] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz. A Reliability-Aware Network Service Chain Provisioning With Delay Guarantees in NFV-Enabled Enterprise Datacenter Networks. *IEEE Transactions on Network and Service Management*, 14(3):554–568, 2017.

[60] 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*, 16(4):1651–1664, 2019.

[61] Andrea Sgambelluri, Alessio Giorgetti, Filippo Cugini, Francesco Paolucci, and Piero Castoldi. OpenFlow-based Segment Protection in Ethernet Networks. *IEEE/OSA Journal of Optical Communications and Networking*, 5:1066–1075, 2013.

[62] C. Natalino, P. Monti, L. França, and title=Dimensioning Optical Clouds with Shared-Path Shared-Computing (SPSC) Protection year=2015 organization=IEEE M. Furdek and L. Wosinska and C. R. Francês and J. W. Costa, booktitle=2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR).

[63] D. Yamada and N. Shinomiya. A Solving Method for Computing and Network Resource Minimization Problem in Service Function Chain against Multiple VNF Failures. In *2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019.

[64] Abolfazl Ghazizadeh, Behzad Akbari, and Mohammad M. Tajiki. Joint Reliability-aware and Cost Efficient Path Allocation and VNF Placement using Sharing Scheme. *ArXiv*, 2019.

[65] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache. A Link Failure Recovery Algorithm for Virtual Network Function chaining. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017.

[66] L. Qu, M. Khabbaz, and C. Assi. Reliability-Aware Service Chaining In Carrier-Grade Softwarized Networks. *IEEE Journal on Selected Areas in Communications*, 36(3):558–573, 2018.

[67] M. T. Beck, J. F. Botero, and K. Samelin. Resilient Allocation of Service Function Chains. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2016.

[68] A. Tomassilli, G. Di Lena, F. Giroire, I. Tahiri, D. Saucez, S. Perennes, T. Turletti, R. Sadykov, F. Vanderbeck, and C. Lac. Poster: Design of Survivable SDN/NFV-Enabled Networks with Bandwidth-Optimal Failure Recovery. In *2019 The International Federation for Information Processing (IFIP) Networking (IFIP Networking)*, pages 1–2, 2019.

[69] D Oliveira, J Crichigno, N Siasi, E Bou-Harb, and N Ghani. Joint Mapping and Routing of Virtual Network Functions for Improved Disaster Recovery Support. In *SoutheastCon 2018*, pages 1–8. IEEE, April 2018.

[70] D. Oliveira, M. Pourvali, J. Crichigno, E. Bou-Harb, M. Rahouti, and N. Ghani. An Efficient Multi-Objective Resiliency Scheme for Routing of Virtual Functions in Failure Scenarios. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, pages 123–129. IEEE, 2019.

[71] N. Siasi, A. Jaesim, A. Aldalbahi, and N. Ghani. Link Failure Recovery in NFV for 5G and Beyond. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 144–148. IEEE, 2019.

[72] Nazli Siasi, Nabeel I Sulieman, and Richard D Gitlin. Ultra-reliable NFV-based 5G Networks Using Diversity and Network Coding. In *2018 IEEE 19th Wireless and Microwave Technology Conference (WAMICON)*, pages 1–4. IEEE, 2018.

[73] Seungik Lee and Myung-Ki Shin. A self-recovery scheme for service function chaining. In *International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2015.

[74] Mohammad Mahdi Tajiki, Mohammad Shojafar, Behzad Akbari, Stefano Salsano, and Mauro Conti. Software defined service function chaining with failure consideration for fog computing. *Concurrency and Computation: Practice and Experience, Wiley*, 31, 2019.

[75] 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. *Computer Networks*, 162, 2018.

[76] M. Tajiki, M. Shojafar, B. Akbari, St. Salsano, and M. Conti. Software Defined Service Function Chaining with Failure Consideration for Fog Computing. *Concurrency and Computation: Practice and Experience*, page 4953, 2018.

[77] H. Huang and S. Guo. Proactive Failure Recovery for NFV in Distributed Edge Computing. *IEEE Communications Magazine*, 57(5):131–137, 2019.

[78] Ryuji Oma, Shigenari Nakamura, Dilawaer Duolikun, Tomoya Enokido, and Makoto Takizawa. Fault-Tolerant Fog Computing Models in the IoT. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 14–25, Cham, 2019. Springer International Publishing.

[79] Umar Ozeer, Xavier Etchevers, Loïc Letondeur, François-Gaël Ottogalli, Gwen Salaün, and Jean-Marc Vincent. Resilience of Stateful IoT Applications in a Dynamic Fog Environment. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, page 332–341. Association for Computing Machinery, 2018.

[80] Abdulaziz Alarifi, Fathi Abdelsamie, and Mohammed Amoon. A fault-Tolerant Aware Scheduling Method for Fog-Cloud Environments. *PloS one*, 14(10), 2019.

[81] V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez, and S. Sánchez-López. Proactive vs Reactive Failure Recovery Assessment in Combined Fog-to-Cloud (F2C) systems. In *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2017.

[82] D. Satria, D. Park, and J. Minho. Recovery for Overloaded Mobile Edge Computing. *Future Generation Computer Systems*, 70:138–147, 2017.

[83] Ashkan Yousefpour, Siddartha Devic, Brian Q Nguyen, Aboudy Kreidieh, Alan Liao, Alexandre M Bayen, and Jason P Jue. Guardians of the Deep Fog: Failure-Resilient DNN Inference from Edge to Cloud. In *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, pages 25–31, 2019.

[84] A. Modarresi and J. P. G. Sterbenz. Toward Resilient Networks with Fog Computing. In *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7, 2017.

[85] K. Wang, Y. Shao, L. Xie, J. Wu, and S. Guo. Adaptive and Fault-Tolerant Data Processing in Healthcare IoT Based on Fog Computing. *IEEE Transactions on Network Science and Engineering*, 7(1):263–273, 2020.

[86] Feng Xu, Tamal Das, Min Peng, and Nasir Ghani. Evaluation of Post-Fault Restoration Strategies in Multi-Domain Networks. *Optical Switching and Networking*, 9(2):147–155, 2012.

Appendix A: Glossary of Terms

5G : Fifth generation

BBU : Baseband unit

CPU : Central processing unit

COTS : Commercial-off-the-shelf

DM : Delay minimization

DM-P : Delay minimization protection

DNS : Domain name service

ETSI : European Telecommunications Standards Institute

F-AP : Fog access point

FF-DM : First-fit delay minimization

FF-DM-P : First-fit delay minimization protection

FF-LM : First-fit load minimization

FF-LM : First-fit load minimization protection

ILP : Integer linear programming

IoT : Internet of things

IP : Internet Protocol

LM : Load minimization

LM-P : Load minimization protection

MANO : Management and network orchestration

MILP : Mixed integer linear programming

MIQCP : Mixed-integer quadratically constrained program

ML : Machine learning

NAT : Network address translation

NC : Network controller

NFV : Network function virtualization

NFVI : Network function virtualization infrastructure

OTA : Over-the-air

PaaS : Platform as-a-service

PGW : Packet data gateway

QoS : Quality of service

RAN : Radio access network

RRH : Remote radio head

SDN : Software defined network

SFC : Service function chain

SFR : Service function request

SFI : Service function instance

SGW : Servicing gateway

SRLG : Shared risk link group

TS : Tabu search

VANET : Vehicular adhoc network

VM : Virtual machine

VNE : Virtual network embedding

VNF : Virtual network function

VNFM : Virtual network function manager

WAN : Wireless local network

Appendix B: Variable Definitions

A list of all variable definitions used in this thesis dissertation is presented.

B.1 Chapter 3

b_r : Bandwidth request for request $\mathbf{x}_r$

B_{kl}^{ij} : Maximum bandwidth of link e_{kl}^{ij}

c_j^i : Free available capacity of j-th fog node in Layer i

$cost(\mathbf{x}_r)$: Cost of mapping request $\mathbf{x}_r$

C_j^i : Maximum resource level of j-th fog node in Layer i

dst_r : Closest primary fog node to Layer 1 destination terminal

D_{prc} : VNF processing delays at mapped nodes

D_{com} : Communication delays in vector path

e_{kl}^{ij} : Link between the k-th node in Layer i and l-th node in Layer j

$\mathbf{E}$: Set of network communication links

$\mathbf{E}^{'}$: Set of pruned links with insufficient resources

f_m : VNF instance

$\mathbf{F}$: Set of VNFs

$\mathbf{F}_r$: Ordered set of requested functions in service chain for request $\mathbf{x}_r$

$\mathbf{G}$: Multi-layer graph

$\mathbf{G}'$: Temporary feasible graph

K : Number of shortest paths

L_{node} : Aggregate node load

L_{link} : Aggregate link load

$\mathbf{N}$: Set of network (fog, cloud) nodes

$\mathbf{N}'$: Set of pruned nodes with insufficient resources

n_j^i : j-th node at Layer i

Δ_{ij}^m : Processing delay for running VNF f_m at node n_j^i

$O(\cdot)$: Computational complexity order

P_m : Resource requirement for VNF f_m

χ_j^i : Unit resource usage cost at node n_j^i

src_r : Closest primary fog node to requesting Layer 1 source terminal

T : Total number of different VNF types

$\mathbf{P}$: Set of path vectors (up to K in total)

$\mathbf{p}_s$: Individual path vector containing path nodes and links

Q : Number of mapping combinations of VNFs on path nodes

$\mathbf{u}_t$: t-th combination vector

u_t^m : Position of m-th VNF mapping in t-th path combination vector, $\mathbf{u}_t$

$\mathbf{v}_t$: Mapping vector of 2-tuple pairs noting mapped path nodes for each VNF

$\mathbf{x}_r$: Single SFC request/demand

$\mathbf{X}_{in}$: Set of input batch SFC requests

$\mathbf{X}'$: Sorted set of batch SFC requests (ordered based on bandwidth or VNF resources)

χ_j^i : Unit resource usage cost at node n_j^i

$\Phi(\cdot)$: Combination generating function

Γ_{kl}^{ij} : Unit bandwidth usage cost at link e_{kl}^{ij}

δ_r : Delay bound threshold for request r

B.2 Chapter 4

L : Path length

$\mathbf{Q}$: Set of node/link-disjoint paths for each path in $\mathbf{P}$

Q_1 : Number of mapping combinations of VNFs on primary path nodes

Q_2 : Number of mapping combinations of VNFs on backup path nodes

$\mathbf{u}_t$: t-th combination vector for primary (working) path

u_t^m : Position of m-th VNF mapping in t-th primary path combination vector, $\mathbf{u}_t$

$\mathbf{v}_t$: Mapping vector of 2-tuple pairs noting mapped primary path nodes for each VNF

$\mathbf{y}_t$: t-th combination vector for backup path

y_t^m : Position of m-th VNF mapping in t-th backup path combination vector, $\mathbf{y}_t$

$\mathbf{w}_t$: Mapping vector of 2-tuple pairs noting mapped backup path nodes for each VNF

B.3 Chapter 5

$\mathbf{E}''$: Set of non-failed links (after removing failed links in $\mathbf{E}$)

$\mathbf{G}''$: Working graph (with failed nodes and links pruned)

$\mathbf{N}''$: Set of non-failed nodes (after removing failed nodes in $\mathbf{N}$)

$\mathbf{X}_{success}$: Set of successfully-provisioned requests from input batch $\mathbf{X}_{in}$

$\mathbf{X}_{fail}$: Set of failed SFC demands in $\mathbf{X}_{success}$

$\mathbf{X}'_{fail}$: Ordered set of failed SFC demands in $\mathbf{X}_{success}$

$\mathbf{X}_{rest}$: Set of successfully-restored SFC demands from $\mathbf{X}_{fail}$

$\mathbf{x}'$: Temporary SFC request (end-to-end or intermediate)

n^- : Failed node

dst'_r : Anterior embedding node to failed node n^-

src'_r : Anterior embedding node to failed node n^-

Appendix C: Copyright Permissions

The permission below is for the use of Figures 2.2, 2.3 and 2.4.

Tabu Search for Efficient Service Function Chain Provisioning in Fog Networks

Conference Proceedings:
2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)

Author: Nazli Siasi

Publisher: IEEE

Date: Dec. 2019

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Container-Based Service Function Chain Mapping

Conference Proceedings: 2019 SoutheastCon

Author: N. Siasi

Publisher: IEEE

Date: April 2019

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

About the Author

Nazli Siasi received her Bachelor and Masters degrees in Information Technology Engineering from Tabriz University of Technology and Tehran Polytechnic University of Technology in 2011 and 2014, respectively. Subsequently she spent several years working in industry before starting her graduate studies in the Department of Electrical Engineering at the University of South Florida under the supervision of Prof. N. Ghani (in 2017). Here she first completed her M.S. degree with focus on Systems & Security and then proceeded to complete her doctoral studies. Furthermore she also did an internship at Cisco (San Jose, CA) in the summer of 2018 and received both the Ali Sharifi Scholarship and a National Science Foundation (NSF) Student Travel Award in 2019. Her research interests include cloud and fog computing, network virtualization, software defined networks, wireless 5G networks, cybersecurity, and machine learning.