Doctoral theses at NTNU, 2020:399

Yordanos Tibebu Woldeyohannes

Efficient Allocation of Resources in NFV-Enabled Networks

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Yordanos Tibebu Woldeyohannes

Efficient Allocation of Resources in NFV-Enabled Networks

Thesis for the Degree of Philosophiae Doctor

Trondheim, December 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

ISBN 978-82-326-5146-7 (printed ver.)
ISBN 978-82-326-5147-4 (electronic ver.)
ISSN 2703-8084 (online)
ISSN 1503-8181 (printed ver.)

Doctoral theses at NTNU, 2020:399

Printed by NTNU Grafisk senter

Abstract

The network traffic and number of devices that are connected within a network are increasing, driven by the increase in network coverage as well as the types of network services provisioned. The provisioning of network services requires different network functions that are often chained in specific order. The traditional implementation of network functions is hardware-based, in which each network function (NF) has its own specific proprietary hardware and software. Besides being costly, this approach has complicated the network management and slowed the pace in which new services are adapted to the market. To address these issues, the networking industry is pursuing Network Function Virtualization (NFV), which enables cost efficient, softwarized and flexible network service provisioning. One of the key challenges in NFV is resource allocation. This thesis work is purposed at addressing the problem of allocating resources efficiently in an NFV-enabled network by striking a balance across multiple objectives and satisfying the performance and availability requirements of service requests. The research contributions of the PhD work are classified into three parts.

First, an efficient NFV resource allocation algorithm called ClusPR is proposed. ClusPR is developed based on observations made from the optimal solution obtained by solving an Integer Linear Programming (ILP) model. ClusPR strikes a balance between multiple objectives including minimizing the path stretch, maximizing the utilization of the network and balancing the load among NF instances. Compared to the state-of-the art approaches, ClusPR is able to satisfy the service performance (i.e., delay) requirement of more flows. In addition, an online algorithm called iClusPR that performs dynamic horizontal scaling by creating and/or removing NF instances depending on the traffic demand is proposed. The performance of iClusPR is close to its offline counterpart ClusPR.

Second, a set of new network structural dependency measures are proposed to overcome a major shortcoming of the existing measures, which is that they do not take into account network fragmentation caused by failure of a node. The proposed measures referred to as *dependency indexes* assess the impact failure of a node has on the information communication between two nodes (path), between one node and the other network nodes (node), and between any one of the remaining network nodes (network) by explicitly considering possible fragmentation. The applicability of the network dependency index for correctly identifying the critical nodes of a network is also demonstrated.

Third, for ensuring high availability of services redundant or backup service chains should be allocated. The NFV redundancy allocation problem is modeled by using two ILP models referred to as AllOne and AllAny. Although the models give the optimal result the execution time of the models increases exponentially with an increase in the size of the network as the problem is NP-hard. To address this problem, a scalable redundancy allocation scheme called CoShare is proposed. CoShare meticulously select backups to avoid the simultaneous unavailability of both the primary and backup chains due to network structural dependencies. In addition, CoShare uses resources efficiently by adopting an approach referred to as *NF shared reservation*, in which the reserved capacity at a backup NF instance is shared among service requests or flows that are not expected to fail simultaneously. The results show that by utilizing NF shared reservation, CoShare is able to reduce the *resource overbuild*, which is measured as the amount of extra capacity (i.e., number of backup NF instances) required as a percentage of the capacity needed without redundancy (i.e., number of primary NF instances).

In summary, this thesis work provides algorithms that find efficient allocation of resources in an NFV-enabled network while ensuring the fulfillment of both the performance and availability requirements of service requests.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree Philosophiae doctor (PhD) at NTNU, Norwegian University of Science and Technology. The PhD thesis work is carried out in the Department of Information Security and Communication Technology (IIK). The PhD was partly financed by the EU FP7 Marie Curie Actions by the EC Seventh Framework Programme (FP7/2007-2013) Grant Agreement No. 607584 (the Cleansky ITN project).

Acknowledgment

Above all I thank God, for all is made possible by his grace. "Thanks be to God for his indescribable gift!" (2 Corinthians 9:15)

It is with deep gratitude that I thank my supervisor Prof. Yuming Jiang, to whom I will forever be grateful for taking a leap of faith in me and entrusting me with the PhD position. He has been tremendously supportive and understanding throughout the PhD work, and has facilitated great research collaborations for me. He has encouraged me to explore new research areas and has shaped my research work. I am also very grateful to Prof. K. K. Ramakrishnan, who has played a key role in my thesis work by introducing me to the NFV resource allocation problem and by tirelessly providing his guidance. His immense knowledge and critical views have had an invaluable impact on the PhD research work. It has been such a great privilege and honor to work with Prof. K. K. Ramakrishnan. I would also like to thank Ali Mohammadkhan for the discussions we had during our research collaboration. My great thanks to Besmir Tola for the fruitful collaborations we had as well as for opening my eyes to all the bureaucracy I had overlooked and being a nice office-mate throughout the years.

I would like to thank my hosts during my secondments at UNINETT, Adj. prof Otto Jonassen Wittner, and at Nokia Bell labs, Dr. Volker Hilt, for the knowledge shared as well as for the friendly work environment. I also like to extend my thanks to all the members of the Cleansky ITN, our meeting in Cleansky workshops and conferences have been quite informative. I would also like to thank my colleagues and group-mates at the department of IIK for the nice and friendly work environment especially Mona, Katina, Katrien De Moor, and David.

My special thanks and deep appreciation to my parent, my heroic dad, Tibe, and my sweet mom, Enatye, for their endless love, for always encouraging me to aim high and for believing in me even when I doubt myself. They have led the foundation for who I am today, all this would not have been possible without their love, support and prayer. I would also like to thank my siblings, my lovely sisters Alem, Betty, and brothers Johnny and Abrish, who have always been there for me in thick and thin. I am blessed to have you all.

It is with much love that I thank my dear husband Merkebu and my son Noah, who have first-hand traversed the ups and downs of the PhD journey with me, you two are my world. My husband has been incredibly supportive in every aspects, from handling all daily chores during my travels, proofreading my papers and PhD thesis, to emotional support during my tough times of the PhD work. Mare, I am blessed to have such a good-hearted, thoughtful husband and best friend like you. To my sweet son, my greatest gift of all, you are my sunshine and joy. It has been such a wonderful blessing to watch you grow up.

Dedication

This PhD thesis is dedicated to my mom, the most kind, caring and loving person. Enatye you will forever live in our hearts.

Contents

List of Acronyms

CAPEX	Capital Expenditure
CNP	Critical Node Problem
COTS	Commercial off-the-shelf
DPI	Deep Packet Inspection
ETSI	European Telecommunications Standards Institute
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
MANO	Management and Operation
MILP	Mixed Integer Linear Programming
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator
NFV	Network Function Virtualization
NF	Network Function
NOS	Network Operating System
NSH	Network Service Header
NS	Network Service

OPEX	Operational Expenditure
SDN	Software-Defined Networking
SFC	Service Function Chaining
SLA	Service Level Agreement
SPI	Service Path Identifier
VIM	Virtual Infrastructure Management
VM	Virtual Machine
VNE	Virtual Network Embedding
VNFM	Virtual Network Function Manager
VNF	Virtual Network Function
WAN	Wide Area Network

List of Tables

List of Figures

Part I

Thesis Summary

Chapter 1

Introduction and Research Objectives

1.1 Outline of the Thesis

This thesis is structured in two parts. Part I presents the summary of the thesis and part II contains the papers included in the thesis. Part I is further structured in four chapters. Chapter one includes the introduction of the thesis work, the research questions addressed as well as the list of publications included in the thesis and their relationship with each other. The background and related works are described in Chapter two. In Chapter three, the contributions of the thesis and a summary of the included papers are explained. Finally, Chapter four presents the conclusion and future work.

1.2 Introduction

Network services such as mobile voice/data, video streaming are provisioned by composing different network functions (NFs), which are utilized for various purposes [107, 116], such as improving security (e.g., firewall, Intrusion Detection Systems (IDS)), network performance (e.g., proxies, WAN optimizer), providing value-added services (e.g., parental control) etc. Traditional NFs are hardware-based in which each network functionality is implemented using a purpose-built proprietary hardware and software. The increase in network traffic as well as the type of network services provisioned has necessitated the deployment of a large number of diverse types of NFs in telecommunication networks. In turn, the operation and management of the network is increasingly becoming complex and inflexible due to the hardware-based implementation of the NFs, which also has resulted in high capital expenditure (CAPEX) and operational expenditure (OPEX) [107]. This is because the management tools of NFs vary across different types as well as vendors. In addition, provisioning new network services or expanding service coverage requires the installation and operation of new NF hardware [107, 88].

On the other hand, the network traffic is increasing at a fast rate, with forcasts predicting three fold increase within a period of five years (2017-2022) [24]. In addition, with the realization of 5G being on the horizon, new disruptive and innovative services are envisioned by massive connectivity between people, machines, and things. To support these emerging services telecommunication networks should have features like agility, cost efficiency, as well as programmablity [118], which are features not possessed by the legacy networks developed using hardware-based NFs. To address these problems, the concept of Network Function Vertualization (NFV) was first conceived in 2012 by the European Telecommunications Standards Institute (ETSI) [23].

NFV decouples the software implementation of network functions from the specialized dedicated hardware and runs the NFs in virtualized environment such as virtual machines (VMs) or containers [88, 23]. NFV brings flexibility in the provisioning of services and is envisioned to revolutionize the way telecommunication networks are designed and managed [55]. NFV has attracted enormous attention both from academia and industry. However, despite the momentum, NFV technology is not mature enough yet and network providers are reluctant to adopt it because of a number of challenges [89, 10, 3]. One of the key challenges is resource management and orchestration (MANO) [89, 3].

The NFV resource allocation problem is new and challenging as it incorporates different levels of decision making, which are at a network-level (NF placement) and flow-level (service chaining and routing). In NFV, NF instances are created on the fly depending on the traffic demand and network status. This means, NF instances should be placed optimally on the physical host Commercial-off-the-Shelf (COTS) hardware, the capacity of the NF instances need to be scaled in or out dynamically, and for each network service (NS) request the NF instances should be selected optimally and chained to a service function chain (SFC) of the NS [37, 89]. In addition, customers expect their services to have the required *performance* (e.g delay, throughput) and *dependability* (availability and reliability).

One of the main factors hindering the widespread adoption of NFV is performance. Service providers want the guarentee that NFV will be able to provide the quality of service (in terms of latency, throughtput, etc) which is comparable to the dedicated specialized hardware [88, 10]. Thus, NFV resource allocation schemes should satisfy the performance requirements of users and also use resources efficiently so that the cost saving benefits of NFV are materialized.

Considering the cruciality of the NFV resource allocation problem, a number of works have developed either mathematical models that find exact solutions and/or heuristic algorithms that are more scalable than the models but give sub-optimal solutions. In [57], a comprehensive survey of the literature in NFV resource allocation is provided. However, the reviewed works do not efficiently solve the problem. The first reason is that the NF placement and flow routing decisions are inherently correlated as service chains of flows' are composed during flow routing by utilizing instances placed in NF placement. This interdependency should be taken into account to enable efficient utilization of resources. However, the NF placement and flow routing problems are solved separately without factoring in their interdependency in works like [26, 75, 28, 65, 60]. The second reason is that, resource allocation decisions need to be made by balancing across multiple objectives including minimizing the path stretch (delay), maximizing utilization of the network, so that service providers could guarantee that they are able to fulfill service performance requirements of flows while using resources efficiently to ensure their profitability. However, existing works either minimize path stretch disregarding its effect on the network utilization [105, 103, 13], or maximize the utilization without considering the delay performance [94, 36, 71, 67, 84].

Another main challenge is fulfilling the service availability requirements of flows. NFV-enabled networks are expected to be able to support carrier-grade services that need five nines (99,999%) availability (which is equivalent to 5.26 minutes of downtime per year) or higher [40]. High service availability cannot be achieved by the mere provisioning of a primary service chain [31, 42]. As the COTS servers used in NFV-enabled networks are less reliable compared to the dedicated hardware [53, 52]. In addition, service function chaining also exacerbates the problem as the failure of one of the NFs of a chain will result in service discontinuity [41]. As redundancy is the "de-facto" technique for achieving high availability [15], a redundant or backup NF should be assigned to takeover the service in case the primary instance fail. However, redundancy could become ineffective due to correlated failures that also result in the unavailability of the backup.

To increase the effectiveness of redundancy, backups should be selected meticulously so as to avoid the simultaneous unavailability of both the primary and backup chains. Inherent network stuctural dependencies could result in correlated failures that might undermine the effect of the redundancy [119]. Thus, understanding and quantifying the network structural dependency facilitates the provisioning of robust services. The existing centrality measures quantify the structural dependency in a network by measuring the connectivity degradation that is caused by a removal or failure of a node [74, 66]. However, as will be demonstrated in this thesis, the existing measures have a major shortcoming that is they do not take into account the possibility of network fragmentation after the removal of a node. Thus, there is a need for a new measure that addresses this shortcoming.

In addition, unless planned carefully redundancy can be inefficient and costly in terms of resource usage especially for achieving high availability [42]. The redundancy allocation approaches in the literature have considered restrictive setups that limit the resource utilization effciency. For example, a backup chain is constrained to use NFs hosted on one node in [76, 43], or a backup NF instance serves one flow or single-tenancy in [95, 30, 120]. In traditional IP/MPLS networks sharing of resources in redundancy has been shown to be effective in achieving efficiency [99, 77, 114]. However, in most of the exising works, backup resources are not shared, instead are dedicated to a flow or instance [95, 30, 120, 43]. Thus, the development of an NFV redundancy allocation approach that utilizes resources efficiently by serving multiple flows on one NF instance, constructing a backup chain using NF instances that might also be hosted on different nodes and sharing redundancy resources is an open issue.

Another important aspect that has been disregarded in the existing works is the effect of network structural dependency that could render the redundancy ineffective. For example in [76], a neighboring node of a primary node is selected as a backup without checking if the two nodes are inherently dependent due to the topology. This PhD thesis addresses these gaps by tackling the following research questions,

RQ1. *How to make efficient resource allocation decisions in NFV-enabled networks considering the inter-dependency between NF placement and flow routing decisions and balancing across multiple objectives?* (Paper A and Paper B)

RQ2. *How to measure network structural dependencies by taking into consideration the possibility that failure of a node might result in fragmentation of the network?* (Paper C)

RQ3. *How to perform efficient redundancy allocation considering the effect of network structural dependencies and utilizing resources efficiently?* (Paper D and Paper E)

1.3 Research Objective

The objective of this thesis work is to propose schemes for enabling efficient allocation of resources in NFV-enabled networks by striking a balance across multiple objectives while fulfilling the service performance and availability requirements. Specifically,

- (**O1**): To develop a mathematical model that finds the optimal allocation of resources while guaranteeing that the delay requirements are satisfied by striking a balance between multiple objectives (including path stretch, network utilization and load balancing).
- (**O2**): To make observations from the optimal decision making of the mathematical model, based on which, to propose a scalable algorithm whose performance is close to the optimal.
- (**O3**): To develop an algorithm that dynamically scales the number of NF instances depending on the traffic demand and network status.
- (**O4**): To demonstrate the shortcoming of existing network structural dependence measures and propose new measures that address this issue.
- (**O5**): To develop mathematical models that find optimal allocation of backups for the NFV redundancy allocation problem.
- (**O6**): To develop a scalable algorithm that allocates backup chains, considering the effect of network structural dependencies and utilizing resources efficiently.

1.4 List of Publications

Table 1.1 shows a list of publications included in this thesis work. The relationships of the papers with each other, the research questions and objectives are also depicted in the Fig 1.1. The relationships indicate paper extensions and/or usage of proposed measures, results or algorithms. For example, paper B is an extended version of paper A. In papers D and E the measures proposed in paper C are used. The included papers have been published in peer-reviewed international journals (Paper B, paper C) and conferences (paper D, paper A) while Paper E is available on arXiv at https://arxiv.org/abs/2008.13453, and to be submitted to a journal.

Paper A	**Yordanos Tibebu Woldeyohannes**, Ali Mohammadkhan, K. K. Ramakrishnan, and Yuming Jiang. "*A Scalable Resource Allocation Scheme for NFV: Balancing Utilization and Path Stretch.*" 21st Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, February 2018.
Paper B	**Yordanos Tibebu Woldeyohannes**, Ali Mohammadkhan, K. K. Ramakrishnan, and Yuming Jiang."*ClusPR: Balancing Multiple Objectives at Scale for NFV resource allocation*". IEEE Transactions on Network and Service Management 15, no. 4 (2018): 1307-1321.
Paper C	**Yordanos Tibebu Woldeyohannes**, and Yuming Jiang. "*Measures for Network Structural Dependency Analysis*". IEEE Communications Letters 22.10 (2018): 2052-2055.
Paper D	**Yordanos Tibebu Woldeyohannes**, Besmir Tola, and Yuming Jiang."*Towards Carrier-Grade Service Provisioning in NFV.*" 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal 2019.
Paper E	**Yordanos Tibebu Woldeyohannes**, Besmir Tola, Yuming Jiang, and K. K. Ramakrishnan. "*CoShare: An Efficient Approach for Redundancy Allocation in NFV*", available on arXiv at https://arxiv.org/abs/2008.13453, and to be submitted to a journal.

Table 1.1: List of publications included

Paper F	Mao, Zhifei, Yuming Jiang, Xiaoqiang Di, and **Yordanos Tibebu Woldeyohannes**. "Joint Head Selection and Airtime Allocation for Data Dissemination in Mobile Social Networks. Computer Networks. 2020 Jan 15;166:106990.

Table 1.2: Other publication during the PhD work

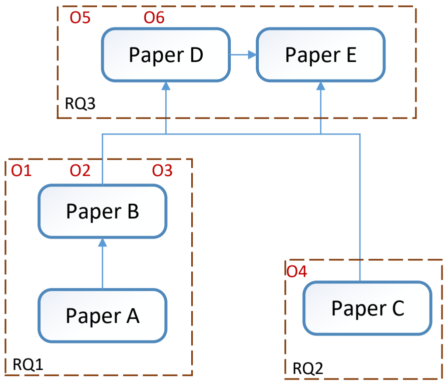


Figure 1.1: Relation of the included papers

Chapter 2

Background and Related Work

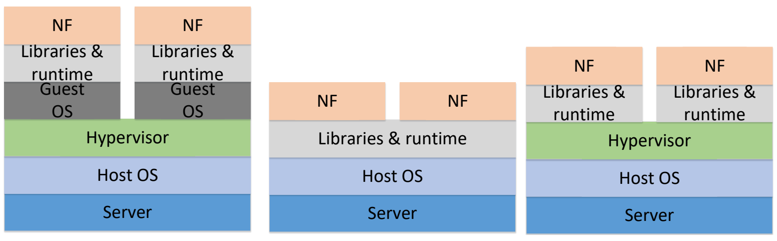
In this chapter, background concepts of the thesis work as well as an overview of the related work to the addressed research questions are presented. This chapter is composed of two sections in which Section 2.1 presents the background by discussing Network Function Virtualization (NFV), service function chaining as well as software defined networking (SDN), and the relationship between SDN and NFV. In Section 2.2, the problems addressed in this thesis are discussed in conjunction with the related works, namely resource allocation in NFV, network structural dependency analysis and service availability in NFV.

2.1 Background

To enable cost efficient, flexible and programmable networks, the networking industry is adverting to *network softwarization*, which is enabled by Network Function Virtualization (NFV) and Software-Defined Networking (SDN) [45]. NFV aims at decoupling the hardware and software of network functions. SDN on the other hand makes the network programmable by separating the control and data plane.

2.1.1 Network Function Virtualization (NFV)

NFV decouples the software implementation of NFs from their specialized dedicated hardware and runs the software on virtualized environment (e.g., virtual machines (VMs) or containers). Figure 2.1 shows NF instances running on virtualized environments that are created using different virtualization technologies. The VMs or contrainers running on the same server are interconnected using vSwitches like Open vSwitch (OvS) [7], VALE [102]. The introduction of high performance packet I/O libraries such as DPDK [2], netmap [101] have enabled the software



(a) Virtual machine **(b)** Container **(c)** Unikernel

Figure 2.1: NF instances running on different virtualizaion technologies [9].

based vSwitches to process packets at the line rate of 10Gbps or higher. Various software dataplane frameworks have been proposed for NFV, based on different virtualization technology and packet I/O [93].

The virtualized environment considered first to be used in NFV is VM [23]. VMs can be created using hypervisor technologies such as KVM [4], Xen [11], VMware. The hypervisors provide isolation between the VMs running on the same server. Thus, the failure of a VM will not affect the other VMs running on the server. Dataplane frameworks that are based on VM include softNIC [54] and NetVM [63], which utilize KVM for virtualization and Intel DPDK for packet I/O. The disadvantage of using VMs to run network functions is their overhead. VMs typically consume large amount of memory in the order of hundreds of megabytes to several gigabytes as they require their own guest OS, so the instantiation of a VM takes time in the order of seconds [79].

Considering this, the usage of container technologies such as dockers [1] and Linux containers [5] is gaining momentum. Containers consume fewer resources (a few MBs or tens of MBs) than virtual machines since they do not include their own operating system, instead relying on the host's kernel [122]. Frameworks that are based on containers include OpenNetVM [122], GNF [29], Flurries [121]. However, containers do not have a strong isolation mechanism making them the target of an ever increasing number of exploits [79]. In addition, any container that can monopolize or exhaust system resources (e.g., memory) will cause a DoS attack on all other containers on that host [79].

The other virutalization technology being considered is unikernel, which is a lightweight or minimalistic virtual machine that provides the mimimum set of libraries required for running a specific application [79]. Unikernel aims to incorporate the high security feature of VM while having an efficiency close to that of containers. ClickOS [82] is an NFV dataplane framework that is based on unikernel. ClickOS runs lightweight virtual machines using Xen hypervisor and mini-OS, customized for network packet processing. It uses netmap [101] and the VALE switch [102] to efficiently move packets between the lightweigth VMs. The downside of ClickOS is its limited flexibility as NFs must be designed within the Click framework's specification and do not run within a standard Linux environment [122].

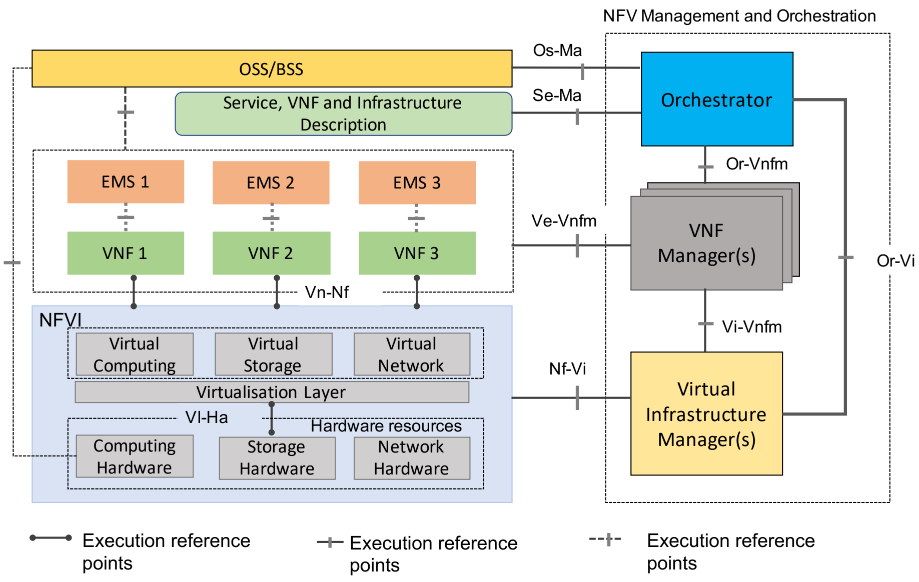


Figure 2.2: NFV reference architectural framework [39].

NFV ETSI Architecture

The European Telecommunications Standards Institute (ETSI) is the standardization body for NFV and defines a reference architecture of an NFV platform [39], which is shown in Fig. 2.2. On a high-level, the architecture consists of three main components: NFV Infrastructure (NFVI), Virtualised Network Functions (VNFs) and NFV Management and Orchestration (MANO) [39]. The components are described briefly below.

NFV Infrastructure (NFVI): NFVI contains all the hardware and software components required for building up the virtualized environment in which NF instances are deployed. This includes the physical resources (such as compute, network, storage), which can span across several locations, virtualization layer and virtual resources.

Virtualised Network Functions (VNFs): Virtual Network Functions (abbreviated as VNFs, or NFs) are the software components that implement the network functions, which will run on a virtualized environment created by the adopted virtualization technology.

NFV Management and Orchestration (MANO): MANO is responsible for the overall control and management of an NFV-enabled network. The MANO is made of three modules each of which has a specific functionality [98]. These include Virtual Infrastructure Management (VIM), Virtual Network Function Manager (VNFM) and NFV Orchestrator (NFVO).

NFV Orchestrator: The NFV Orchestrator is responsible for the creation and management of end-to-end services. It has two main components, resource orchestration and service orchestration. The resource orchestration is purposed at supporting the service delivery by managing NFVI resources, which in turn are controlled by one or more VIMs. While service orchestration is targeted at life cycle management of network services.

Virtual Network Function Manager (VNFM): The VNFM is responsible for the lifecycle management of NF instances. The specific functions of VNFM include NF instance creation, modification, scaling out/in and up/down, NF configuration (if required), performance and fault management. Each NF instance needs to have a VNFM, which may also manage other NF instances as well.

Virtual Infrastructure Management (VIM): The Virtual Infrastructure Management (VIM) is responsible for managing the resources in the NFVI, including physical resources (compute, storage and network), virtual resources (VMs) and software resources (hypervisors), which are usually within one operator's infrastructure domain [98].

2.1.2 Service Function Chaining

The delivery of an end-to-end service (e.g. mobile voice/data, Internet access, a virtual private network) usually requires multiple network functions, which often need to be interconnected in a specific order. Service Function Chaining (SFC) is the definition and instantiation of an ordered list of NF instances and the subsequent steering of traffic flows through those NFs [97]. For example, Fig 2.3 shows two flows that require different service function chains. SFC presents a model addressing the problematic aspects of existing service deployments, including topological dependence and configuration complexity [51].

The IETF has proposed Network Service Header (NSH) [96], which is a data-plane protocol, to enable service function chaining. The NSH facilitates the forwarding of packets between service functions or NFs of a service chain by creating a dedicated service plane which is independent of the underlying transport protocol. The

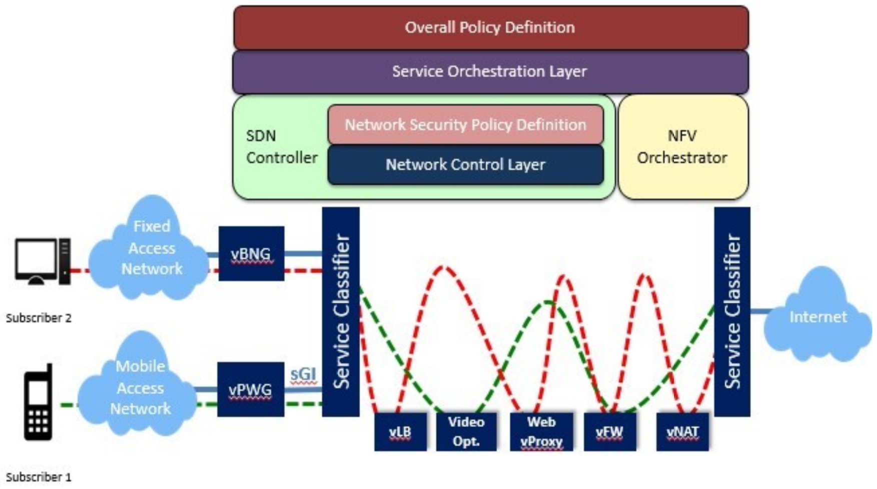


Figure 2.3: Service function chaining example [8].

NSH consists of a mandatory base header, service path header and context header. The base header consists of information like version, TTL, flags. The service path header contains Service Path Identifier (SPI), which specifies where packets assigned to a service path must go, and service index, which provides the location within a service path. The context header is optional and carries the metadata information. NSH is appended to a packet/frame and an outer transport encapsulation (e.g., MPLS, VXLAN) is imposed on the NSH.

The NSH is topology independent and offers a common and standards-based header for service chaining to all network and service nodes. Furthermore, the NSH provides service-specific Operations, Administration, and Maintenance (OAM) messages which are useful for monitoring and troubleshooting a service chain [96].

2.1.3 Software-Defined Networking (SDN)

In traditional IP networks, networking devices perform both control plane tasks of deciding how to handle network traffic as well as the data plane functionality of forwarding packets. This tight coupling between the control and data planes has resulted in a static architecture that is complex to manage and control. In addition, it has hindered innovation and evolution of the networking infrastructure, as the design, evaluation and deployment of new protocols takes years [70].

Software-Defined Networking (SDN) alters the vertically-integrated architecture of the current networks by decoupling the control plane from the data plane. The

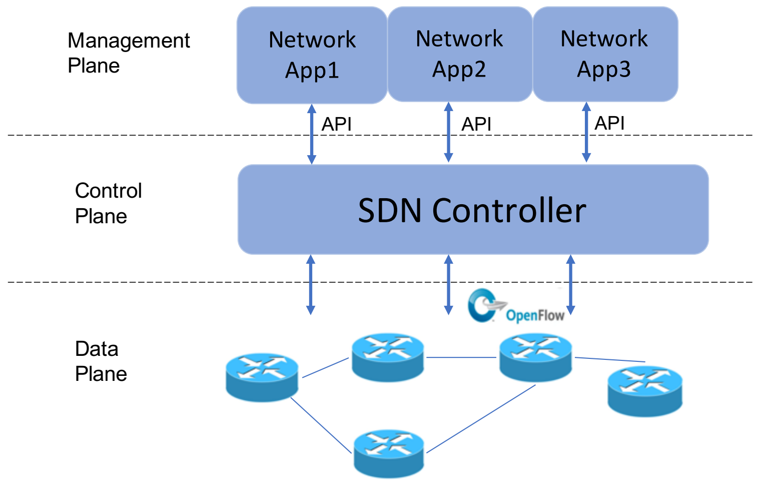


Figure 2.4: Software-defined networking [106].

control plane functionalities will be managed by a logically-centralized controller called the SDN controller or the Network Operating System (NOS). The SDN controller communicates with the data plane forwarding devices through the southbound interface using APIs such as Openflow [83]. The SDN controller will have a network-wide view of the network topology and is responsible for making all routing and control policies. The switches will simply be packet forwarding devices based on the rules stated in their flow tables, the contents of which are decided by the SDN controller. SDN makes the network programmable by offering APIs, which application developers can use to create different network applications. That is application developers interact with the network operating system through northbound interface using the northbound APIs like REST API, and programming languages like Pyretic [100], Procera [113]. Typically, a northbound interface abstracts the low level instruction sets used by southbound interfaces to program forwarding devices [70].

SDN adoption has until now mainly focused on data-center (DC) networks and intra-domain wide area networks (WANs) [87, 70]. For example, Google is using B4 [64], its software-defined inter-DC WAN solution, to interconnect datacenters that are geographically distributed. The centralized control plane approach of SDN has enabled B4 to achieve 100% utilization level on many of the links of the network. Microsoft uses an SDN based solution called SWAN (software-driven WAN), for inter-data center commmunication [59]. SWAN is also able to achieve high level of utilization.

In comparison, inter-domain WANs still heavily rely on legacy routing and traffic engineering technologies [87]. In inter-domain networks, autonomous systems are operated by different parties with different network equipment, policies, and business objectives that cannot be shared with other networks [87]. This complicates the process of applying a logically centralized control across multiple administrative domains. Software Defined Internet Exchange Point (SDX), which is an SDN based Internet Exchange Point (IXP) where network operators exchange traffic and route information, has been proposed to enable SDN based inter-domain WAN [69, 49].

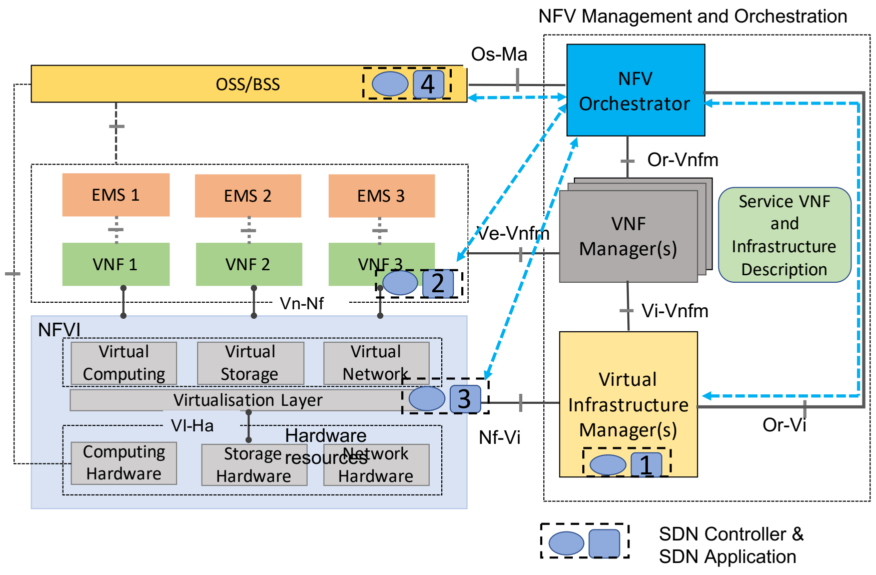


Figure 2.5: SDN in NFV reference architectural framework [38].

Relationship Between NFV and SDN

Software-defined networking (SDN) and network function virtualization (NFV) are enabling technologies for the softwarization of the telecommunication network. SDN and NFV can be implemented and used independently. However, if applied together they highly complement each other. NFV can benefit from the functionalities provided by SDN and vise-versa [88]. The SDN controller is not necessarily a standalone physical entity, which could for example be a software component which runs on a VM or container (i.e., an NF instances). Thus, SDN can make use of the elasticity, dynamicity and agility that NFV provides for running the SDN controller. On the other hand, SDN can be used to steer the network traffic through a chain of NF instances in an NFV-enabled network. For example, the OpenDaylight SDN controller [6] presents a service chaining solution built on top of Software Defined Networking (SDN) using network service header (NSH).

ETSI has defined an SDN integrated NFV reference framework, which specifies possible locations for the SDN controller, applications and resources (physical/virtual routers and switches) in the NFV framework [38], which is shown in Fig 2.5. The classical location of the SDN controller is the NFVI, other possible choices are as part of the VIM, VNF or the OSS. The data plane forwarding components which the SDN controller manages might be physical routers/switches or virtual switches (vSwitches). The SDN controller will communicate with the NFV orchestrator through the orchestration interface to exchange information like the network topology [38].

2.2 Related Work

In this section, the state-of-the-art in the reviewed literature is presented focusing on the problems addressed in this thesis.

2.2.1 Resource Allocation in NFV

To enable flexible NF orchestration, resource allocation decisions need to be made both at network-level (NF placement) and flow-level (flow routing). The Network-level decisions include: (1) finding the optimal number of NF instances, and (2) placing the instances at optimal locations. The flow-level decisions consist of: (3) selecting the NF instances that will serve the flows, and (4) finding optimal routing paths. The NF placement and flow routing decisions are inter-dependent since flows will be assigned instances that are instantiated during NF placement. Thus, for efficient resource allocation, all the four decisions need to be made holistically.

The NFV resource allocation problem has some similarities with the virtual network embedding problem (VNE), which is a problem of allocating resources to virtual network requests onto a shared substrate network i.e., the physical network. VNE deals with the allocation of virtual resources both in nodes (virtual nodes mapped to physical nodes) and links (virtual links mapped to paths connecting the virtual nodes) [44]. The NFs in a service chain can be regarded as the virtual nodes in VNE. However, there are important differences between the two problems [16], which include,

1. *NF instance sharing*: In NFV resource allocation the virtual nodes, i.e., the NF instances, are shared among multiple flows. That is an NF instance might be

used by multiple service chains [16, 86]. However, in VNE virtual nodes in different virtual networks are independent.

2. *NF ordering in a chain*: In NFV, a flow's traffic has to traverse through the NFs of the service chain in a specific sequence as required by the network service [16, 123]. For example, traffic might have to traverse through a firewall before going through a load balancer. However, the VNE problem does not impose this kind of constraint.

Therefore, the NFV resource allocation and VNE problems are different and need to be dealt with accordingly.

In addition, the NFV resource allocation decisions have to be made taking into account the inherent constraints on network resources while fulfilling the service performance and availability requirements. Service providers have a limited set of resources, which include bandwidth of links and computing as well as memory of network nodes. Thus, capacity constraints of resources should not be violated and also resources should be used *efficiently*. Therefore, the NFV resource allocation decisions need to be made taking into account multiple objectives. In this thesis, the following identified factors will be focused on for efficient resource allocation in an NFV-enabled network.

- (P.1): The inter-dependency between network-level (NF placement) and flow-level (flow routing) problems has to be captured in the resource allocation decision making.
- (P.2): An NF instance should be shared by multiple flows, whose service chains include the NF type of the instance, at the maximum possible extent.
- (P.3): The precedence constraints among NFs of a service chain need to be taken into account. Disregarding this in the NF selection process might result in the elongation of the path of the flow.
- (P.4): It is important to strike a balance between multiple objectives that could at times be in conflict with each other.

In summary, the NFV resource allocation problem is a new and challenging problem. Considering the paramount importance of NFV resource allocation various approaches have been proposed in the literature, which include both exact approaches as well as heuristics and metaheuristics. However, the reviewed approaches disregard one or multiple of the factors that are required for efficient allocation of resources as listed above. Survey of the existing approaches can be found in [57, 117]. Below we give a review of the most related and recent literature.

Exact approaches

Mathematical programming methods such as Integer Linear Programming (ILP) and Mixed ILP (MILP) are commonly used by the exact approaches, which find the optimal solutions [12, 78, 90, 111, 104, 91, 47, 62, 92, 86, 26, 65, 60, 75, 28, 109]. The models are solved by using solvers such as CPLEX, LINGO, GLPK, Gurobi [117]. Some of the existing models do not consider one or more of the factors listed above (P.1 - P.4) in their resource allocation. For example, the models in [12, 78, 90, 111, 109] do not fulfill (P.3) as they do not route flows keeping the ordering constraint among NFs of their service chains. Some other models do not capture (P.1) as they solve either the NF placement problem [26, 75, 28] or the flow routing problem [65, 60]. More recently, comprehensive models have been proposed in [47, 62, 92].

However, the NFV resource allocation problem is NP-hard [86, 103]. Therefore, the run time of solving the models increases exponentially with an increase in the size of the network or number of flow requests. Efforts have been made to increase the scalablity of solving the models in [62, 92, 109]. However, the performance achieved even after these improvements still fails short of what is demanded in real systems. For example, the model in [62] can only be used in networks that do not have more than 50 nodes and takes tens of minutes to allocate resources, which is not acceptable in real time systems. To address the scalability problem that is associated with the exact approaches, heuristic and meta heuristic approaches have been proposed.

Heuristic and metaheuristic approaches

A number of heuristic approaches have been proposed in the literature. They can be divided into three groups. The first group of algorithms solve either the NF placement or the flow routing problem only (i.e., not considering P.1). Heuristics for the NF placement have been proposed in [25, 26, 19, 21, 75, 28] and flow routing algorithms are proposed in [65, 61, 35, 85, 60]. These algorithms do not capture the inter-dependency between the NF placement and flow routing decisions as they solve them separately.

The second group of algorithms solve the NFV resource allocation problem as a virtual network embedding (VNE) problem in which the network or graph to be embedded is the service chain request with the NFs being the virtual nodes [115, 84, 34, 78, 18]. These algorithms create NF instances for each of the admitted requests so NF instances are not being shared by multiple flows, not satisfying P.2.

As a result, these algorithms are inefficient in their allocation of resources.

In the final group are algorithms that do not take into account the inherent trade-off across multiple objectives that need to be addressed for efficient allocation of resources in NFV-enabled networks. For example, the resource allocation algorithm in [105], [103], [33] and centrality-based heuristics in [21] and [13] avoid path stretch by serving the flows utilizing NF instances placed on its shortest path disregarding its effect on network utilization. In CoMb [105], a flow is constrained to use NF instances running in the same node that is found on its path. However, the CoMb approach can considerably limit the utilization of the network since flows are constrained to stay on their path and use a single node for all their services. Relatively, the centrality-based heuristic in [13] has a relaxed restriction as it allows a flow to use NFs placed on more than one node, but still the nodes have to be located on the shortest path of the flow.

On the other hand, algorithms proposed in [94, 36, 71, 67, 84] try to increase the utilization of the network disregarding the path stretch. In [71], an algorithm that tries to make better use of network resources by promoting flows to reuse instances which have been created instead of instantiating new ones is proposed. In [36], the objective of consolidating NF instances to reduce the number of running host servers is considered. However, they do not consider the effect this will have on the performance of the flows and the SLA satisfaction. Another heuristic approach is the E2 framework [94] which is developed for allocating NF instances and routing flows inside a central office or small data centers. The placement is modeled as a graph partitioning problem and solved using a modified Kernighan-Lin heuristic. Flows are assigned to NFs balancing load across the NF instances.

Thus, there is a need for a resource allocation scheme that enables efficient utilization of resources by considering all the four points, i.e., P.1 to P.4, which leads us to the first research question

RQ1. *How to make efficient network-level and flow-level resource allocation decisions considering their inter-dependency and balancing across multiple objectives ?*

2.2.2 Network Structural Dependency

Due to the inherent structural dependency in a network, the impact of a node's failure on the connectivity of the other network nodes varies from node to node. For example, in a star topology, in which all the nodes except the central node have only one link that is connected to the central node, failure of the central node will result in a connectivity loss between all nodes of the network while failure of any other node won't have an effect on the network connectivity of the others. Here arises an important question, what measures may be used to assess the network structural dependency-caused impact?

The impact of a node's failure might be analyzed at different levels: (1) at a path level on information communication from one node to another node in the network, (2) at the node level on information communication from one node to other nodes in the network and (3) at the network level on information communication among any nodes in the network.

A number of graph-theoretic or centrality measures have been introduced for quantifying the importance of a node in a network. The classical centrality measures are degree, closeness and betweeness [46]. A brief description of these measures is given below. For the purpose of this, a network which is represented as an undirected graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of N nodes and $\mathcal{L}$ is the set of L links of the network, with each link having a weight of one, is considered. The shortest path distance, also known as the geodesic, between nodes i and j is represented as d_{ij}.

The *degree centrality* measures the importance of a node by the number of ties it has with the other network nodes. The degree centrality of node i is given as

$$C_i^D = \sum_{j \in \mathcal{N}} a_{ij} \tag{2.1}$$

where $a_{ij} = 1$ if there is a link between nodes i and j, zero otherwise. Thus, according to the degree centrality, the more the number of links of a node the more important the node is.

The *closeness centrality* [17] quantifies the importance of a node by how near it is with the other nodes of the network. It is measured based on the geodesic. The closeness centrality of node i is given as

$$C_i^C = \frac{N-1}{\sum_{j \in \mathcal{N}} d_{ij}}, \tag{2.2}$$

where d_{ij} is the geodesic between nodes i and j. The inverse of C_i^C, i.e., $\frac{\sum_{j \in \mathcal{N}} d_{ij}}{N-1}$, is the average distance from node i to all the other nodes. Thus, the smallest the average distance the more important a node is based on the closeness centrality.

The *betweenness centrality* assesses the importance of a node by the number of shortest paths that pass through the node. The betweenness centrality of node i is given as

$$C_k^B = \sum_i \sum_j \frac{g_{ikj}}{g_{ij}}, \tag{2.3}$$

where g_{ij} is the number of geodesic between nodes i and j, and g_{ikj} is the number of geodesic between nodes i and j which pass through node k. So based on the betweenness centrality measure, the more the number of routes that go through a node the more important the node is.

In addition, a number of variants of these classical measures have been proposed in the literature [20]. However, these measures are not suitable for the structural dependency-impact problem. This is because their definition of importance of a node is based on prominence and reachability [14], not considering connectivity, which is the base for the network structural dependency analysis.

Existing network structural dependency measures

In [74], a new class of centrality measures called Delta centrality are introduced. Delta centrality measures assess the importance of a node in relation to the ability of a network to respond to the deactivation of the node from the network. These measures quantify the structural dependency at the network level, i.e., the effect failure of a node will have on the information communication between any nodes of the network. A type of delta centrality, referred to as information centrality, is also proposed in [74]. The information centrality is given as

$$C_i^I = \frac{E[G] - E[G^{-i}]}{E[G]} \tag{2.4}$$

where G^{-i} is graph G with node i removed. $E[G]$ and $E[G^{-i}]$ denote the efficiency of the network G and G^{-i} respectively. The network efficiency, which is initially introduced in [73] based on the geodesic, is defined as:

$$E[G] = \frac{1}{N(N-1)} \sum_{\forall i \neq j \in \mathcal{N}} \frac{1}{d_{ij}}. \tag{2.5}$$

Information centrality measures the structural dependency only at the network level, i.e., the impact failure of a node will have on the connectivity among the rest of the network nodes.

In [66], measures that quantify the structural dependency at path and node levels are introduced. The path level measure, denoted as $D(i \to j|n)$, measures the impact of a node n on the path from node i to node j. It is defined as

$$D(i \to j|n) = \frac{1}{d_{ij}} - \frac{1}{d_{ij}^{-n}}, \tag{2.6}$$

where d_{ij}^{-n} denotes the geodesic distance between nodes i and j in the network G^{-n}. By the definitions, it is clear that $d_{ij}^{-n} \geq d_{ij}$. Also in [66], based on the path level measure $D(i \rightarrow j|n)$, a node level measure, denoted as $D(i|n)$, is introduced. $D(i|n)$ measures the average influence or impact of node n on node i and is defined as:

$$D(i|n) \quad = \quad \frac{1}{N-1} \sum_{j \in \mathcal{N}^{-n}} D(i \rightarrow j|n), \tag{2.7}$$

where the set $\mathcal{N}^{-n}$ is equivalent with the set of nodes $\mathcal{N}$ with node n removed. By applying equation (2.6), it can be further written as

$$D(i|n) \quad = \quad \frac{1}{N-1} \sum_{j \in \mathcal{N}^{-n}} (\frac{1}{d_{ij}} - \frac{1}{d_{ij}^{-n}}). \tag{2.8}$$

However, the existing dependency measures have a major shortcoming, which is they do not take into account network fragmentation that failure of a node might cause. This limits their applicability and even correctness in analyzing the network structural dependency. Thus, there is a need for new network structural dependency measures that takes into explicit consideration if network fragmentation due to failure or removal of a node.

Critical Node Detection

Critical nodes of a network are nodes whose removal significantly degrades network connectivity [72]. That is removal of the critical nodes results in the maximum network fragmentation [14]. The classical centrality measures like degree, closeness and betweenness assess the importance of a node based on only its characteristics [14, 48], without considering the effect its removal will have on the other nodes. A comparative study in [48] has demonstrated that the criticality of a node in a network cannot be effectively assessed independent of evaluating the connectivity between the other remaining nodes in the network. Thus, these classical measures are not suitable for finding the critical nodes of a network [14, 32].

In [14], an optimization based approach for identifying critical nodes was introduced. Given a graph and an integer k, the objective of the critical node problem (CNP) is to find a set of k nodes in the graph whose deletion results in the maximum network fragmentation [14]. Since then a number of variants of the CNP problem have been proposed in the literature. A recent survey provides a review of these works [72]. However, the optimization model approach has shortcomings the first of which is scalability. The CNP problem has been proven to be NP-complete [14], thus the model can not be used for large-scale networks. Although heuristics have been proposed for addressing this problem [14], the heuristics are sub-optimal. The second problem is that the optimization model does not quantify the criticality level of the nodes it identifies. The number of critical nodes to be determined is given to the algorithm as an input. However, depending on the structure of the network under consideration failure of the critical nodes might have different levels of impact, so the number of critical nodes is generally a function of the network type.

In [48], it is shown that the impact of removal or failure of a node is dependent on the spatial structure of the network, not necessarily on the characteristics of individual nodes or arcs. Network-level structural dependency measures quantify the effect that removal of a node will have on information communication among the other nodes of the network. Thus, the critical node detection problem has similarities with the network structural dependency analysis problem. However, the existing network structural dependency measures do not factor in fragmentation [74], thus they are also not suitable for identifying the critical nodes of a network [14, 32]. To address this issue, the following research question is formulated in this thesis work.

RQ2. How to measure network structural dependencies at path, node and the network levels by taking into consideration the possibility of fragmentation of the network?

2.2.3 Service Availability in NFV

NFV-enabled networks should be able to support services that have diverse levels of availability requirements ranging from best-effort services like data transfer to real time communication based services that demand high-levels of availability [40]. Redundancy based fault tolerance strategy is often required for fulfilling the availability requirement of services as primary service chains might not have the required availability levels [31, 42]. Considering this, redundancy allocation schemes both exact [112, 58] as well as based on heuristics [42, 56, 43, 41, 30] have been proposed for the NFV redundancy allocation problem.

In [58] three ILP models which allocate service chains that are resilient against single-node/link, single-link and single-node failures. A scalable exact method based on a decomposition model using column generation is proposed for allocating link-disjoint backup path in [112]. However, both [112, 58] do not verify if the backup chain allocated is able to satisfy the service availability requirement. Thus, these schemes are not able to guarantee that the services provisioned to users are in accordance with their availability requirements.

A greedy algorithm that allocates off-site backup NFs by iteratively allocating backup to the least available NF of a service chain is proposed in [41]. A Cost-Importance Measure (CIM) is used for selecting the NF instance to be provisioned a backup instead of the least available NF in [30]. In [120] backup NFs are allocated with the purpose of minimizing the backup resource consumption while considering the heterogeneity of VNF resource requirements. The schemes in [30, 41, 120] consider cloud services that require upto 99% or 99.9% service availability. The authors of [41] extend their work in [42] considering heterogeneous availability requirements including carrier-grade services that demand 99.999% availability with the objective of maximizing the number of service chain requests served. The adoption of simplified assumptions such as all the backup nodes have the same availability [120, 56], considering only the failure of NFs and disregarding physical nodes, or assuming NF instances fail independently irrespective of their placement [42] limit the applicability of these works.

Redundancy can be costly so efficient utilization of resources is crucial in backup allocation. In traditional IP/MPLS networks sharing of resources in redundancy has been shown to be effective in achieving efficiency [99, 77, 114]. In [95] computing resources on the same physical host machine are allowed to be shared by adjacent VNF backups, but, a backup NF instance serves one flow or single-tenancy, as in [95, 30, 120]. In [76], a multi-tenancy based approach that allows multiple flows to be served by a backup NF instance is proposed and shown to outperforms single-tenancy based approaches. However, in [76], a backup chain is constrained to only use NFs hosted on one node, similar to [43], which limits the resource utilization efficiency as backup chains cannot be constructed by utilizing NFs from different host nodes there.

Another important aspect that has been disregarded in the existing works is the effect network structural/topological dependencies have on the redundancy. Unless planned carefully, network structural dependencies could lead to the failure of both the primary and backup chains, undermining the effect of the redundancy. For instance, in [76], a neighboring node of a primary node is selected as a backup without checking if the two nodes are inherently dependent due to the topology. These trigger the following research question:

RQ3. How to perform redundancy allocation considering the effect of network structural dependencies and utilizing resources efficiently?

Chapter 3

Research Methodology and Contributions

In this chapter, the research methodology followed in the thesis work is explained first. Then, the research contributions made are highlighted followed by a brief summary of the included papers.

3.1 Research Methodology

The research methodology adopted in this thesis work follows the standard scientific research process [68], and is outlined in Fig. 3.1. For a research problem, the related literature is first studied extensively, which leads to the development of the research questions. Following that, the system model and assumptions are defined. In this thesis work, the research questions are addressed by proposing measures, mathematical optimization models and/or heuristic algorithms based on the research questions being tackled.

Mathematical optimization

A mathematical optimization problem, or just an optimization problem, has the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq b_i, \ ; i = 1, \ldots, m. \end{aligned} \tag{3.1}$$

where the vector $\mathbf{x} = (x_1, \ldots, x_n)$ is the optimization variable of the problem, the function $f_0(x) : R^n \longrightarrow R$ is the objective function, while the functions $f_i(x) : R^n \longrightarrow R, \ i = 1, \ldots, m$, are the inequality constraint functions, and the constants $b_i, \ i = 1, \ldots, m$, are the limits, or bounds, for the constraints [22].

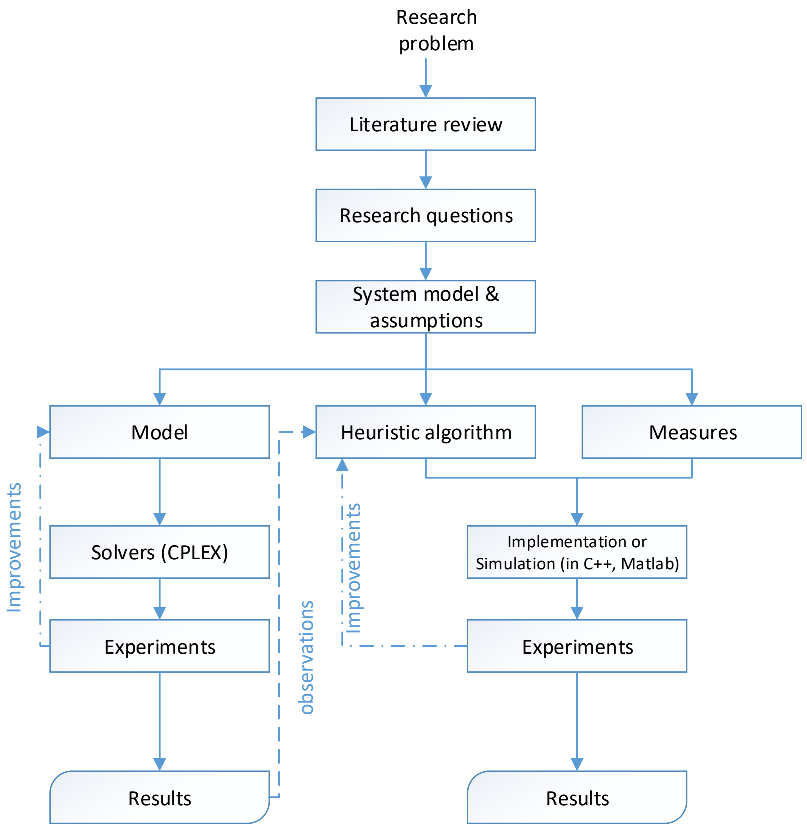


Figure 3.1: Research methodology

A vector $\mathbf{x}^*$ is called *optimal*, or a solution of the problem, if it has the smallest objective value among all vectors that satisfy the constraints, for any feasible z i.e $f_i(z) \leq b_i$, for all $i, \ldots, m$ and $f_0(z) >= f_0(\mathbf{x}^*)$. There are different classes of optimization problems based on the form of the objection and constraint functions. For example, if the objective and constraint functions are all linear, the optimization problem is referred to as linear program. In Quadratic programming optimization problems the objective function is a quadratic function and the constraint functions are linear. Convex optimization problems are a class of optimization problems whereby the objective and constraint functions are convex. Convex optimization can also be defined as optimizing a convex function over a convex set.

Definition 1 (Convex set) A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta x_1 + (1-\theta)x_2 \in C \tag{3.2}$$

Definition 2 (Convex function) A function $f : R^n \rightarrow R$ is convex, if for every $x, y \in R^n$ and $0 \leq \theta \leq 1$ the inequality (3.3) is satisfied.

$$f\left(\theta x + (1-\theta)y\right) \leq \theta f(x) + (1-\theta)f(y) \tag{3.3}$$

Convex optimization encompasses different optimization classes including least square, linear programming, geometric programming e.t.c,. The complexity of an optimization problem depends on factors such as the class of the optimization problem (i.e., the form of the objective and constraint functions), and the size of the problem (i.e., the number of variables and constraint functions). The optimal solution of convex optimization problem can be found in polynomial time, even for large-sized problems, using effective algorithms such as interior-point method, sub-gradient method, [22].

Another class of optimization problems is discrete optimization in which some or all of the variables are restricted to be discrete. Integer linear programming (ILP) is a type of discrete optimization in which the variables are integers and the objective and constraint functions are linear. Algorithms such as cutting plane methods and branch and bound are able to find exact solutions of ILP problems for small-sized problems [108]. Optimization solvers such as CPLEX [27] and Gurobi [50] have implementation of these algorithms. ILP problems are generally NP-hard, so the run time of the algorithms increases exponentially with an increase in the size of the network. Thus, heuristic algorithms are required for solving most practical ILP problems

NFV resource allocation problem is modeled as an ILP (Integer Linear Program). A number of experiments are conducted on a small network by varying different parameters. The purpose of these experiments is to get insight into the optimal decision making. Based on these insights, heuristic algorithms that are scalable and give close to optimal solutions are developed.

Heuristic algorithms

The observations obtained from solving the mathematical models are used as input for developing scalable heuristic algorithms. The proposed algorithms are implemented using C++ and Matlab. Several experiments are conducted to validate and test the performance of the algorithms. The test results are used to further improve the algorithms. In addition, the results of the algorithms are compared with the optimal solutions of the models for small networks. This is done to test how close the results of the heuristic algorithms are to the optimal solutions.

When required, besides the implementation of the tests, simulations are conducted to test the performance of algorithms. The simulation settings used are close to real systems in which flows arrive to a system randomly, get service for a random amount of time and depart. The performance of the models is analyzed by using realistic network topologies such as the Rocketfuel network topology [110]. In addition, the performance of the algorithms is compared with the existing state-of-the art algorithms.

Measures

For the network structural dependency analysis, measures are developed. The performance of the measures is compared with the existing measures using different network topologies.

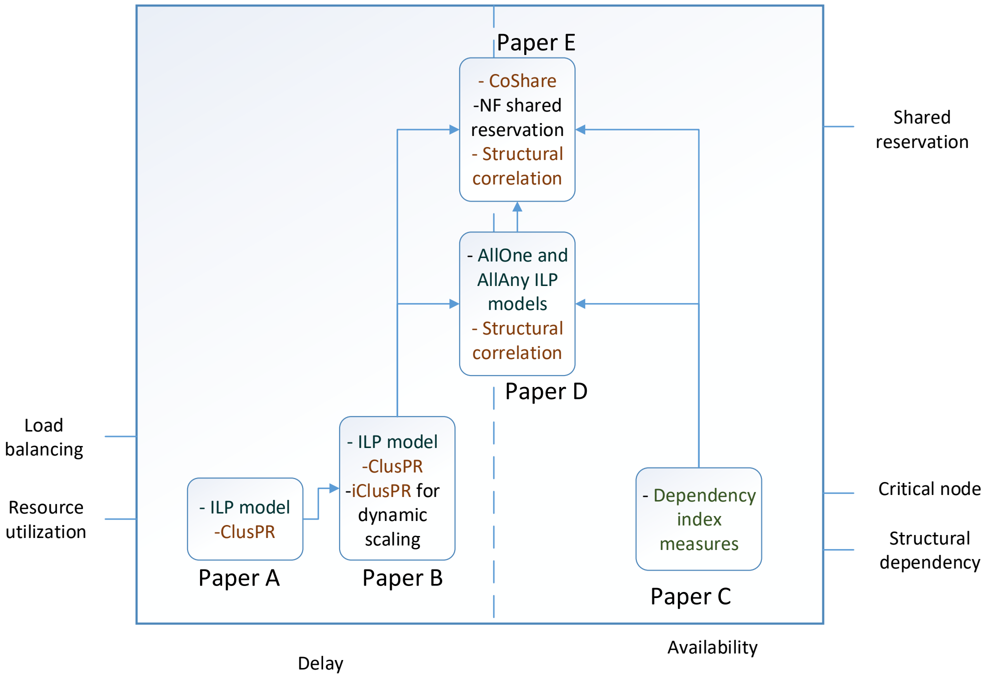


Figure 3.2: Main research contributions in relation to the included papers

3.2 Research Contributions

In this thesis work, six main contributions are made. Fig 3.2 highlights the main contributions in relation to the included papers. The contributions are explained in brief below,

- **Contribution 1** : The first contribution of this thesis is an Integer Linear Programming *(ILP) model* that optimally allocates resources for NFV-enabled networks (papers A and B). The model is comprehensive as it solves the NF placement and flow routing problems holistically thereby capturing their inter-dependency. It also ensures that the delay performance requirements of flows are met, the ordering constraint among NF of a service chain is kept and the capacity constraints of major resources such as, processing, memory, bandwidth are not violated. In addition, insightful observations are made by solving the proposed model on different settings.

- **Contribution 2** : The observations from **Contribution 1** are used as input for developing a scalable *resource allocation algorithm* called *ClusPR* (paper A and B). ClusPR strikes a balance across multiple objectives (minimizing path stretch (delay), maximizing total utilization of the network and balancing the load among NF instances). ClusPR captures the inter-dependency between NF placement and flow routing decisions by utilizing a new approach that is based on clustering. This approach helps balance between the objectives of minimizing path stretch and maximizing network utilization. A novel flow routing algorithm that is based on dynamic programming is used to strike a balance between minimizing the path stretch and balancing the load among NF instances in Paper B. Experimental results show that ClusPR outperforms the state-of-the art approaches it is compared with.

- **Contribution 3** : The third contribution is a *dynamic scaling algorithm* referred as *iClusPR* (paper B). iClusPR adjusts the number of NF instances in the network depending on the traffic demand and network status. iClusPR is developed based on the principle of ClusPR so it also inherits the performance advantages of ClusPR. The performance of iClusPR is analyzed in a simulation setting that resembles a real system setting.

- **Contribution 4** : A set of new *measures called dependency indexes* that quantify the network structural dependencies at different levels (i.e., path, node and network) are the fourth contribution. The path, node and network dependency indexes assess the dependency between a node, and a path, another node and the network respectively. The novelty of the measures is their ability to take into explicit consideration the possibility of network fragmentation upon removal or failure of a node. The usability of the network dependency index in identifying critical nodes of a network is also demonstrated in paper C. In addition, the node dependency index is used in papers D and E to identify nodes that have strong structural correlation with a given node, i.e., identify nodes that are likely to become unavailable following failure of the node due to their network structural dependency.

- **Contribution 5** : To fulfill the high-availability requirement of carrier-grade service, which could reach five nines (0.99999) or higher, two *ILP models* referred to as *AllOne* and *AllAny* are proposed for the optimal assignment of redundant or backup chains in paper D. The proposed models apply chain level redundancy where they assign backup NF instances for each of the NFs of a primary chain. The difference between them is that in the AllOne model each flow uses one backup node for all of the NFs of its chain while the AllAny model is more relaxed as the backup NFs of a chain could be hosted on different backup nodes.
- **Contribution 6** : A scalable and efficient backup chain allocation algorithm called *CoShare* is proposed in paper E. To utilize resources efficiently in CoShare, a backup NF instance serves more than one flows, a backup chain can be constructed by utilizing NF instances hosted on different nodes and an approach called *NF shared reservation*, in which the same reserved capacity at a backup NF instance can be shared among flows that are not susceptible to simultaneous failures, is employed. CoShare also assigns backups meticulously to avoid the unavailability of both the primary and backup chains due to correlated unavailability caused by network structural dependency. In addition, CoShare also ensures that the backup chain delay as well as availability requirements of flows are met.

  These six contributions are purposed at addressing the three research questions of the PhD work. Specifically, contributions 1-3 answer the first research question (RQ1), contribution 4 addresses the second research question (RQ2). The third research question (RQ3) is tackled by contributions 5 and 6.

3.3 Summary of the Included Papers

In this section, a summary of the papers included in the thesis is given.

Publication A

Yordanos Tibebu Woldeyohannes, Ali Mohammadkhan, K. K. Ramakrishnan, and Yuming Jiang. "*A Scalable Resource Allocation Scheme for NFV: Balancing Utilization and Path Stretch*." 21st Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, February 2018.

The NFV resource allocation problem is addressed in this paper. First a comprehensive multiple objective MILP model is proposed for the problem. The objectives of the model are maximizing the number of flows admitted, minimizing the utilization of the nodes and links with the purpose of leaving spare resources for future incoming flows. The model solves both the NF placement and flow routing (i.e., assignment of NF instances and end-to-end routing while keeping the ordering) problems simultaneously, thus factoring in their inter-dependency. In addition, besides capacity constraints on the resources (processing, memory, bandwidth), flows' performance constraints in terms of delay are incorporated in the model. The model is solved using CPLEX for different kinds of setting (like varying chain length, network capacity). By analyzing the solutions, four insightful observations are made.

Based on the four observations, a scalable NFV resource allocation algorithm called ClusPR is proposed to address the scalability problem of the model, which is caused by the NP-hardness of the NFV resource allocation problem. ClusPR uses a divide-and-conquer approach and solves the NF placement and flow routing problems sequentially. To capture the inter-dependency between the two problems, a novel approach that is based on clustering is adopted. This approach enables ClusPR to create a balance between the objectives of maximizing the utilization of the network and minimizing the path stretch, which is the additional delay from the shortest path delay. The state-of-the-art approaches in the literature either maximize the utilization of network disregarding its effect on the path stretch or delay, or avoid path stretch by placing the NF instances on the shortest path of flows without considering the utilization of the network. In comparison to the state-of-the-art approaches, ClusPR is able to decrease the average delay by $1.2\times - 1.6\times$ and the worst-case delay by more than $10\times$, while admitting the same or slightly larger number of flows and satisfying the delay requirement of 25%-35% more flows.

Publication B

Yordanos Tibebu Woldeyohannes, Ali Mohammadkhan, K. K. Ramakrishnan, and Yuming Jiang."*ClusPR: Balancing Multiple Objectives at Scale for NFV resource allocation*". IEEE Transactions on Network and Service Management 15, no. 4 (2018): 1307-1321.

.

In paper B, besides the objectives of maximizing the utilization of resources and minimizing the path stretch, load balancing is incorporated in ClusPR. A novel flow routing algorithm that is based on dynamic programming is proposed to balance the load among NF instances while minimizing the path stretch. Simulation results demonstrate that the proposed algorithm is able to balance the load and also that load balancing has an added advantage of increasing the number of flows whose delay requirement is satisfied.

One of the advantages of using NFV is the flexibility of creating instances dynamically depending on the traffic demand and network status. Considering this, an online dynamic scaling algorithm, referred to as iClusPR, is proposed. iClusPR inherits the architecture of ClusPR so it also shares its properties like minimizing path stretch and load balancing. iClusPR uses a threshold based approach to do horizontal scaling of the NF instances (i.e., increase/decrease the number of NF instances) in a network based on the traffic demand. The performance of iClusPR is analyzed by using an experimental setting that resembles a real system in which flows arrive randomly, get serviced for a random amount of time and depart.

Publication C

Yordanos Tibebu Woldeyohannes, and Yuming Jiang. "*Measures for Network Structural Dependency Analysis*". IEEE Communications Letters 22.10 (2018): 2052-2055.

In this paper, we demonstrate that the existing network structural dependency measures have a major shortcoming, which is that they do not consider network fragmentation inflicted by failure of a node. The effect this shortcoming has on the values and correctness of the structural analysis is demonstrated by making use of an example. In addition, the unification of the existing measures is also proven. A new set of network structural dependency analysis measures, called dependency indexes, which consider fragmentation that might be caused by failure of a node are proposed to address the problem. The proposed measures assess the dependency at the path level on information communication between two nodes (*path dependency index*), at the node level on information communication from any node in the network (*node dependency index*), and at the network level on information communication between any one of the network nodes (*network dependency index*). It is also proven that the values of the network dependency index are a function of the level of fragmentation that the failure of a node will cause.

In addition, the usefulness of the network dependency index in solving the critical node detection problem is demonstrated. The performance of the proposed measures is assessed on different types and sizes of networks ranging from small to large scale. The results show that the critical nodes identified by network dependency index are more in line with intuition compared to other measures like node degree.

Publication D

Yordanos Tibebu Woldeyohannes, Besmir Tola, and Yuming Jiang.“*Towards Carrier-Grade Service Provisioning in NFV*.” 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal 2019.

In paper D, the problem of guaranteeing service availability in NFV is addressed. The “de-facto” method for boosting and ensuring service availability is through redundancy. Considering this, redundancy allocation schemes are proposed. The schemes place backup NF instances and assign them to flows depending on their availability requirement. However, unless planned carefully, network structural dependencies could make the redundancy ineffective by causing the unavailability of the backup as well. Thus, a primary NF instance should not have a strong structural correlation with its backup counterpart. In paper D, an algorithm which identifies network nodes that have high level of structural correlation is proposed. The algorithm uses the node dependency index, proposed in paper C, to quantify the level of dependency between a pair of network nodes.

To tolerate correlated unavailability that is caused by network structural dependencies, flows are not assigned backup NF instances which are hosted on backup nodes having strong structural correlation with a primary node of the flow. The redundancy allocation schemes, referred to as AllOne and AllAny, utilize the proposed algorithm for identifying nodes that have high level of structural correlation. Both AllOne and AllAny models assign chain level redundancy that is all the NFs of a primary chain are assigned a backup. Their difference is that in the AllOne model all the backup NFs of a flow are hosted on one backup node, while in the AllAny model the backup NFs of a flow might be hosted on one or more backup nodes. The models are solved by using CPLEX. The results show that not considering the structural dependency affects the effectiveness of the redundancy allocation effort. When the structural correlation is not taken into account during the redundancy allocation, many of the flows that are ideally expected to have five nines (99.999%) availability may actually only receive two nines (99%) or three nines (99.9%) availability.

Publication E

Yordanos Tibebu Woldeyohannes, Besmir Tola, Yuming Jiang, and K. K. Ramakrishnan. “*CoShare: An Efficient Approach for Redundancy Allocation in NFV*”, available on arXiv at https://arxiv.org/abs/2008.13453, and to be submitted to a journal.

Although the AllOne and AllAny models proposed in paper D give optimal results, they are not scalable for large scale networks. To address this problem, a scalable redundancy allocation scheme called CoShare is proposed. To avert the effect of network structural dependencies that might lead to the unavailability of both the primary and backup chains, CoShare meticulously assigns backup chains by considering the structural dependency among nodes. For this purpose, CoShare utilizes an algorithm that identifies structurally correlated nodes using the network dependency index measure. For utilizing resources efficiently, CoShare takes advantage of the flexibility benefits of NFV by constructing a backup chain utilizing NF instances that are potentially hosted on different nodes and serving multiple flows at each NF instance.

CoShare also enables the sharing of backup resources by adopting an approach called *NF shared reservation*. In NF shared reservation, the reserved processing capacity of an NF instance can be shared among multiple flows whose primary chains are not susceptible to simultaneous failures. Service chains that do not have a common node, whose flows are referred to as independent flows, are not expected to fail simultaneously upon failure of a single node. In NF shared reservation, the corresponding flows can share a reserved processing capacity at an NF instance provided that the flows have the NF type in their service chains. In this case, the instance will reserve a capacity that is enough for only one of the independent flows. CoShare incorporates NF shared reservation while allocating backup service chains to flows. The results show that by adopting NF shared reservation, CoShare is able to significantly decrease the *resource overbuild* (i.e., the number of backup NF instances as a percentage of the number of primary NF instances).

Chapter 4

Conclusion and Future Work

4.1 Conclusion

The traditional approach of having a dedicated hardware device designed to perform a specific network functionality has proven to be costly and inflexible, slowing the innovation of new network services. NFV alters this rigid architecture by decoupling the software implementation of NFs from the dedicated hardware, and instead runs the NF software on a virtualized environment (e.g., VMs or containers) created on top of general-purpose hardware. Thus, different types of NFs can be consolidated into a given hardware and NF instances can be created and/or removed dynamically. Thus, NFV brings flexibility in the management and provisioning of network services. However, NFV also introduces new resource allocation problems that are of paramount importance for actualizing the flexibility. This thesis addresses the problem of allocating resources efficiently in an NFV-enabled network by striking a balance across multiple objectives while fulfilling the service performance and availability requirements of flows. In summary, the solutions proposed for addressing this problem are:

- An efficient NFV resource allocation algorithm called ClusPR, which is developed based on observations made in solving an ILP model proposed for finding the optimal solution. Compared to the state-of-the art approaches, ClusPR is able to satisfy the service performance requirement of more flows. An online algorithm called iClusPR that performs horizontal scaling is also proposed.

- A set of new network structural dependency measures, called dependency indexes, are proposed to overcome a major shortcoming of the existing measures, which is that they do not take into account the possibility of network fragmentation inflicted by failure of a node. In addition, the usability of the measures in identifying critical nodes of a network and in finding nodes that have strong structural correlation is demonstrated.

- An efficient redundancy allocation scheme referred to as CoShare is proposed to guarantee the service availability requirements of flows. CoShare efficiently utilizes resources by adopting a new approach called NF shared reservation, which enables the sharing of reserved processing capacity at an NF instance among flows whose primary chains are not susceptible to simultaneous failures.

4.1.1 Discussion on Generality

The measures and algorithms proposed in this thesis are applicable directly or in modified form on other related problems. For instance, the dynamic programming based routing algorithm of ClusPR can be utilized for allocating resources by balancing across multiple objectives in Virtual Network Embedding (VNE) [44], that is the problem of embedding virtual networks in substrate networks. The CoShare algorithm might also be adapted and used to enable efficient utilization of resources for the redundancy allocation problem in VNE. The proposed dependency index measures, on the other hand, could be utilized to analyze the path, node and network level dependency among nodes of a network from different application areas.

4.2 Future Work

As future work, the following research directions are worth exploring.

Learning algorithms: An interesting problem is to explore the applicability of learning algorithms, which are based on machine learning and artificial intelligence, for the NFV resource allocation problem. Inspired by recent advances in deep reinforcement learning for AI problems, the usage of deep reinforcement learning (DRL) for resource management problems is gaining attraction [80]. For example, DRL has been used to develop adaptive bitrate algorithms for video players [81]. It will be interesting to explore the possibility of developing learning algorithms for the NFV resource allocation problem.

NF vertical scaling: Dynamic scaling of NF instances includes horizontal scaling (i.e., creating or removing instances) and vertical scaling (i.e., increasing or decreasing the amount of resources assigned to an NF instance). However, in this thesis work only horizontal scaling is addressed. Thus, it will be interesting to study further vertical scaling approaches.

Implementation: The performance of the proposed heuristics are analyzed using simulations. As a future work, it will be interesting to implement and evaluate the algorithms in a real test bed.

Bibliography

[1] Docker. https://www.docker.com/. Accessed: 2019-01-19.

[2] Dpdk. https://www.dpdk.org/. Accessed: 2019-01-19.

[3] How to meet the challenges of nfv orchestration. https://searchnetworking.techtarget.com/tip/How-to-meet-the-challenges-of-NFV-orchestration. Accessed: 2019-03-06.

[4] KVM. https://www.linux-kvm.org/. Accessed: 2019-01-19.

[5] LXC. https://linuxcontainers.org/. Accessed: 2019-01-19.

[6] Opendaylight. https://www.opendaylight.org/. Accessed: 2019-01-19.

[7] Openvswitch. http://www.openvswitch.org/. Accessed: 2019-01-19.

[8] Service function chain. https://www.reply.com/en/topics/architecture/introduce-dynamic-service-chaining-by-using-sdn-and-nfv-technolo
Accessed: 2019-01-19.

[9] Unikernels meet NFV. https://www.ericsson.com/en/blog/2016/6/unikernels-meet-nfv. Accessed: 2019-01-19.

[10] Virtual versus reality: The challenges of enterprise nfv adoption. https://www.sdxcentral.com/articles/contributed/the-challenges-of-enterprise-nfv-adoption/2017/10/. Accessed: 2019-03-06.

[11] Xen. https://xenproject.org/. Accessed: 2019-01-19.

[12] B. Addis, D. Belabed, M. Bouet, and S. Secci. Virtual network functions placement and routing optimization. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pages 171–177. IEEE, 2015.

[13] S. Ahvar, H. P. Phyu, S. M. Buddhacharya, E. Ahvar, N. Crespi, and R. Glitho. CCVP: Cost-efficient centrality-based vnf placement and chaining algorithm for network service provisioning. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9. IEEE, 2017.

[14] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[15] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33, 2004.

[16] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. On orchestrating virtual network functions. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 50–56. IEEE, 2015.

[17] A. Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.

[18] M. T. Beck and J. F. Botero. Coordinated allocation of service function chains. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[19] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan. Optimal virtual network function placement in multi-cloud service function chaining architecture. *Computer Communications*, 102:1–16, 2017.

[20] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

[21] M. Bouet, J. Leguay, T. Combe, and V. Conan. Cost-based placement of vdpi functions in nfv infrastructures. *International Journal of Network Management*, 25(6):490–506, 2015.

[22] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[23] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress*, volume 48. sn, 2012.

[24] V. Cisco. Cisco visual networking index: Forecast and trends, 2017–2022. *White Paper*, 1, 2018.

[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)*, pages 1–9. IEEE, 2014.

[26] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz. Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354. IEEE, 2015.

[27] I. I. Cplex. V12. 1: UserâĂŹs manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[28] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros. Dynamic, latency-optimal vnf placement at the network edge. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 693–701. IEEE, 2018.

[29] R. Cziva and D. P. Pezaros. Container network functions: bringing nfv to the network edge. *IEEE Communications Magazine*, 55(6):24–31, 2017.

[30] W. Ding, H. Yu, and S. Luo. Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[31] N.-T. Dinh and Y. Kim. An efficient availability guaranteed deployment scheme for iot service chains over fog-core cloud networks. *Sensors*, 18(11):3970, 2018.

[32] T. N. Dinh, Y. Xuan, M. T. Thai, P. M. Pardalos, and T. Znati. On new approaches of assessing network vulnerability: hardness and approximation. *IEEE/ACM Transactions on Networking*, 20(2):609–619, 2012.

[33] S. Dräxler and H. Karl. Specification, composition, and placement of network services with flexible structures. *International Journal of Network Management*, 27(2):e1963, 2017.

[34] S. Dräxler, H. Karl, and Z. A. Mann. Joint optimization of scaling and placement of virtual network services. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 365–370. IEEE Press, 2017.

[35] A. Dwaraki and T. Wolf. Adaptive service-chain routing for virtual network functions in software-defined networks. In *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, pages 32–37. ACM, 2016.

[36] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca. An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Transactions on Networking*, 25(4):2008–2025, 2017.

[37] M. Ersue. Etsi nfv management and orchestration-an overview. In *Proc. of 88th IETF meeting*, 2013.

[38] I. ETSI. Network functions virtualisation (nfv); ecosystem; report on sdn usage in NFV architectural framework. *ETSI GS NFV-EVE*, 5:V1, 2015.

[39] I. N. ETSI. ETSI GS NFV 002 v1. 1.1: Network Functions Virtualisation (NFV); Architectural Framework, 2013.

[40] I. N. ETSI. ETSI GS NFV-REL 001 v1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements, 2015.

[41] J. Fan, C. Guan, Y. Zhao, and C. Qiao. Availability-aware mapping of service function chains. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[42] J. Fan, M. Jiang, and C. Qiao. Carrier-grade availability-aware mapping of service function chains with on-site backups. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2017.

[43] J. Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao. A framework for provisioning availability of nfv in data center networks. *IEEE Journal on Selected Areas in Communications*, 36(10):2246–2259, 2018.

[44] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[45] H. Freeman and R. Boutaba. Networking ind ustry transf ormation through softwarization [the president's page]. *IEEE Communications Magazine*, 54(8):4–6, 2016.

[46] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[47] M. Gao, B. Addis, M. Bouet, and S. Secci. Optimal orchestration of virtual network functions. *Computer Networks*, 142:108–127, 2018.

[48] T. H. Grubesic, T. C. Matisziw, A. T. Murray, and D. Snediker. Comparative approaches for assessing network vulnerability. *International Regional Science Review*, 31(1):88–112, 2008.

[49] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever. An industrial-scale software defined internet exchange point. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 1–14, 2016.

[50] I. Gurobi Optimization. Gurobi optimizer reference manual. *URL http://www. gurobi. com*, 2018.

[51] J. Halpern and C. Pignataro. Service function chaining (sfc) architecture. Technical report, 2015.

[52] 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.

[53] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh. On the resiliency of virtual network functions. *IEEE Communications Magazine*, 55(7):152–157, 2017.

[54] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy. Softnic: A software nic to augment hardware. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-155*, 2015.

[55] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal. Nfv: State of the art, challenges and implementation in next generation mobile networks (vepc). *arXiv preprint arXiv:1409.4149*, 2014.

[56] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter. Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements. In *Globecom Workshops (GC Wkshps), 2015 IEEE*, pages 1–7. IEEE, 2015.

[57] J. G. Herrera and J. F. Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[58] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina. Virtual network function placement for resilient service chain provisioning. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 245–252. IEEE, 2016.

[59] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven wan. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 15–26. ACM, 2013.

[60] P. Hong, K. Xue, D. Li, et al. Resource aware routing for service function chains in sdn and nfv-enabled network. *IEEE Transactions on Services Computing*, 2018.

[61] H. Huang, S. Guo, J. Wu, and J. Li. Joint middlebox selection and routing for software-defined networking. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[62] N. Huin, B. Jaumard, and F. Giroire. Optimal network service chain provisioning. *IEEE/ACM Transactions on Networking*, 26(3):1320–1333, 2018.

[63] J. Hwang, K. K. Ramakrishnan, and T. Wood. Netvm: High performance and flexible networking using virtualization on commodity platforms. *IEEE Transactions on Network and Service Management*, 12(1):34–47, 2015.

[64] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a globally-deployed software defined wan. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 3–14. ACM, 2013.

[65] S. Jiao, X. Zhang, S. Yu, X. Song, and Z. Xu. Joint virtual network function selection and traffic steering in telecom networks. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE, 2017.

[66] D. Y. Kenett, T. Preis, G. Gur-Gershgoren, and E. Ben-Jacob. Dependency network and node influence: Application to the study of financial markets. *International Journal of Bifurcation and Chaos*, 22(07):1250181, 2012.

[67] S. Khebbache, M. Hadji, and D. Zeghlache. Virtualized network functions chaining and routing algorithms. *Computer Networks*, 114:95–110, 2017.

[68] C. R. Kothari. *Research methodology: Methods and techniques*. New Age International, 2004.

[69] V. Kotronis, X. Dimitropoulos, and B. Ager. Outsourcing the routing control logic: better internet routing based on sdn principles. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 55–60. ACM, 2012.

[70] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[71] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai. Deploying chains of virtual network functions: On the relation between link and server usage. *IEEE/ACM Transactions on Networking (TON)*, 26(4):1562–1576, 2018.

[72] M. Lalou, M. A. Tahraoui, and H. Kheddouci. The critical node detection problem in networks: a survey. *Computer Science Review*, 28:92–117, 2018.

[73] V. Latora and M. Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.

[74] V. Latora and M. Marchiori. A measure of centrality based on network efficiency. *New Journal of Physics*, 9(6):188, 2007.

[75] D. Li, P. Hong, K. Xue, et al. Virtual network function placement considering resource optimization and sfc requests in cloud datacenter. *IEEE Transactions on Parallel and Distributed Systems*, 29(7):1664–1677, 2018.

[76] 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*, 2019.

[77] G. Li, D. Wang, C. Kalmanek, and R. Doverspike. Efficient distributed restoration path selection for shared mesh restoration. *IEEE/ACM Transactions on Networking (TON)*, 11(5):761–771, 2003.

[78] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106. IEEE, 2015.

[79] F. Manco, C. Lupu, F. Schmidt, J. Mendes, S. Kuenzer, S. Sati, K. Yasukata, C. Raiciu, and F. Huici. My vm is lighter (and safer) than your container. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 218–233. ACM, 2017.

[80] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2016.

[81] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210. ACM, 2017.

[82] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. Clickos and the art of network function virtualization. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 459–473, 2014.

[83] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[84] M. Mechtri, C. Ghribi, and D. Zeghlache. A scalable algorithm for the placement of service function chains. *IEEE Transactions on Network and Service Management*, 13(3):533–546, 2016.

[85] A. M. Medhat, G. Carella, C. Lück, M.-I. Corici, and T. Magedanz. Near optimal service function path instantiation in a multi-datacenter environment. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 336–341. IEEE, 2015.

[86] S. Mehraghdam, M. Keller, and H. Karl. Specifying and placing chains of virtual network functions. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 7–13. IEEE, 2014.

[87] O. Michel and E. Keller. Sdn in wide-area networks: A survey. In *2017 Fourth International Conference on Software Defined Systems (SDS)*, pages 37–42. IEEE, 2017.

[88] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[89] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latré, M. Charalambides, and D. Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105, 2016.

[90] H. Moens and F. De Turck. Vnf-p: A model for efficient placement of virtualized network functions. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 418–423. IEEE, 2014.

[91] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. Ramakrishnan, and T. Wood. Virtual function placement and traffic steering in flexible and dynamic software defined networks. In *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*, pages 1–6. IEEE, 2015.

[92] T.-M. Nguyen, M. Minoux, and S. Fdida. Optimizing resource utilization in nfv dynamic systems: New exact and heuristic approaches. *Computer Networks*, 148:129–141, 2019.

[93] Z. Niu, H. Xu, L. Liu, Y. Tian, P. Wang, and Z. Li. Unveiling performance of nfv software dataplanes. In *Proceedings of the 2nd Workshop on Cloud-Assisted Networking*, pages 13–18. ACM, 2017.

[94] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: a framework for NFV applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 121–136. ACM, 2015.

[95] 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.

[96] P. Quinn, U. Elzur, and C. Pignataro. Network service header (nsh). Technical report, 2018.

[97] P. Quinn and T. Nadeau. Problem statement for service function chaining. Technical report, 2015.

[98] J. Quittek, P. Bauskar, T. BenMeriem, A. Bennett, M. Besson, and A. Et. Network functions virtualisation (nfv)-management and orchestration. *ETSI NFV ISG, White Paper*, 2014.

[99] S. Ramamurthy and B. Mukherjee. Survivable wdm mesh networks. part i-protection. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE*

Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320), volume 2, pages 744–751. IEEE, 1999.

[100] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker. Modular sdn programming with pyretic. *Technical Reprot of USENIX*, 2013.

[101] L. Rizzo. Netmap: a novel framework for fast packet i/o. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112, 2012.

[102] L. Rizzo and G. Lettieri. Vale, a switched ethernet for virtual machines. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 61–72. ACM, 2012.

[103] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye. Provably efficient algorithms for joint placement and allocation of virtual network functions. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[104] M. Savi, M. Tornatore, and G. Verticale. Impact of processing costs on service chain placement in network functions virtualization. In *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 191–197. IEEE, 2015.

[105] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pages 323–336, 2012.

[106] N. Shah, P. Giaccone, D. B. Rawat, A. Rayes, and N. Zhao. Solutions for adopting software defined network in practice. *International Journal of Communication Systems*, page e3990.

[107] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24, 2012.

[108] G. Sierksma. *Linear and integer programming: theory and practice*. CRC Press, 2001.

[109] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache. Online and batch algorithms for vnfs placement and chaining. *Computer Networks*, 2019.

[110] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[111] M. M. Tajiki, S. Salsano, M. Shojafar, L. Chiaraviglio, and B. Akbari. Energy-efficient path allocation heuristic for service function chaining. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–8. IEEE, 2018.

[112] A. Tomassilli, N. Huin, F. Giroire, and B. Jaumard. Resource requirements for reliable service function chaining. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[113] A. Voellmy, H. Kim, and N. Feamster. Procera: a language for high-level reactive network control. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 43–48. ACM, 2012.

[114] K. Walkowiak and M. Klinkowski. Shared backup path protection in elastic optical networks: Modeling and optimization. In *2013 9th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 187–194. IEEE, 2013.

[115] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta. Joint optimization of service function chaining and resource allocation in network function virtualization. *IEEE Access*, 4:8084–8094, 2016.

[116] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 374–385. ACM, 2011.

[117] B. Yi, X. Wang, K. Li, M. Huang, et al. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018.

[118] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider. Nfv and sdnâĂŤkey technology enablers for 5g networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478, 2017.

[119] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford. Heading off correlated failures through independence-as-a-service. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 317–334, 2014.

[120] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu. Raba: Resource-aware backup allocation for a chain of virtual network functions.

In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1918–1926. IEEE, 2019.

[121] W. Zhang, J. Hwang, S. Rajagopalan, K. Ramakrishnan, and T. Wood. Flurries: Countless fine-grained nfs for flexible per-flow customization. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 3–17. ACM, 2016.

[122] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. Ramakrishnan, and T. Wood. Opennetvm: A platform for high performance network service chains. In *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, pages 26–31. ACM, 2016.

[123] Y. Zhang, C. Truchan, M. Tatipamula, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, et al. Steering: A software-defined networking for inline service chaining. In *2013 21st IEEE international conference on network protocols (ICNP)*, pages 1–10. IEEE, 2013.

Part II

Included Papers

Paper A

A Scalable Resource Allocation Scheme for NFV: Balancing Utilization and Path Stretch

Yordanos T. Woldeyohannes, Ali Mohammadkhan, K. K. Ramakrishnan, and Yuming Jiang

In Proceedings of the 21th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, February 2018.

A Scalable Resource Allocation Scheme for NFV: Balancing Utilization and Path Stretch

Y.T. Woldeyohannes*, Ali Mohammadkhan†, K.K. Ramakrishnan†, Yuming Jiang*
*Norwegian University of Science and Technology, NTNU, Trondheim, Norway
†University of California Riverside, California, USA

***Abstract*—Network Function Virtualization (NFV) implements network middlebox functions in software, enabling them to be more flexible and dynamic. NFV resource allocation methods can exploit the capabilities of virtualization to dynamically instantiate network functions (NFs) to adapt to network conditions and demand. Deploying NFs requires decisions for both NF placement and routing of flows through these NFs in accordance with the required sequence of NFs that process each flow. The challenge in developing NFV resource allocation schemes is the need to manage the dependency between flow-level (routing) and network-level (placement) decisions.**

We model the NFV resource allocation problem as a multi-objective mixed integer linear programming problem, solving both flow-level and network-level decisions simultaneously. The optimal solution is capable of providing placement and routing decisions at a small scale. Based on the learnings from the optimal solution, we develop ClusPR, a heuristic solution that can scale to larger, more practical network environments supporting a larger number of flows. By elegantly capturing the dependency between flow routing and NF placement, ClusPR strikes a balance between minimizing path stretch and maximizing network utilization. Our experiments show ClusPR is capable of achieving near-optimal solution for a large sized network, in an acceptable time. Compared to state-of-the-art approaches, ClusPR is able to decrease the average normalized delay by a factor of $1.2 - 1.6\times$ and the worst-case delay by $9 - 10\times$, with the same or slightly better network utilization.

I. Introduction

Middleboxes such as firewalls, VPN gateways, proxies, intrusion detection and prevention systems, etc., play a central role in today's Internet by providing network resident functionality that examines and potentially modifies the end-to-end traffic flow [1]. Implementation of network resident functionality is gradually migrating to software platforms, providing additional flexibility and extensibility for the capabilities of the network compared to purpose-built hardware appliances. Evolving the network's capabilities can thus involve lower capital expenditures as the software can run on commercial off-the-shelf (COTS) hardware. Network Function Virtualization (NFV) decouples the software of network functions from the physical machine and runs it on virtual machines, or more recently on "containers" [2]. This also brings greater flexibility in resource management as instances of the network functions (NFs) can be created dynamically, and the capacity for a particular function can be scaled up or down depending on traffic demand. Sequences of NFs are common, and the overall service provided by the network by such sequences of NFs is termed "service function chaining" [3].

A. The Challenge

Resource management of NF service chains continues to be a challenge because of the complexity involved. An NFV resource allocation (NFV-RA) mechanism has to make decisions at multiple levels to ensure resources are properly utilized, while performance requirements of flows are met. Resource allocation decisions have to be made both at the network-level and at the flow-level. At the network-level, the allocation algorithm determines the number of NF instances to instantiate in the network to process the flows. In addition, the algorithm needs to determine the placement of the NFs or the physical machines that should host the NFs. However, this network-level decision making has to be coupled with the decision making at the flow-level, as the flow has to be sequenced through the NF instances to form the desired service chain. The flow-level decision making includes the determination of the route for the flow based on the service chain requirement and the choice of the NF instances initiated in the network.

Suboptimal decisions, resulting in NFs placed on network nodes not along the shortest path will result in "path stretch", contributing to increased latency for the flow. In addition, it is important to use resources efficiently and avoid over provisioning of resources so as to maximize the utilization of the network. Finally, it is critical to consider both link capacity and node processing capacity (in terms of CPU cores) when making the resource allocation decisions. The *interdependence between network-level (placement, instantiating the requisite number of NFs) and flow-level (routing) decisions makes the NFV-RA problem new and challenging*, warranting the recent attention it has received in the literature.

There have been a number of papers on NFV-RA published in the recent past. A detailed survey on existing NFV-RA approaches can be found in [4]. Some have focused on NF placement alone [5], [6]. Some recent works have tried to solve NF placement and flow routing jointly. One group of works seeks to minimize the path stretch (delay) while limiting the utilization of the network [7], [8] and thereby limiting capacity. Another seeks to increase capacity by allowing the network utilization to increase, but at the expense of higher delay [9], [10]. Additionally, some of the existing approaches consider only the node capacity constraint (e.g. the MILP formulation in [11] and [12]), ignoring link capacity constraints.

In general, these literature approaches have only addressed part of the NFV-RA problem, far from solving it completely, which demands balancing between path stretch and utilization, factoring all the key resources, and concurrently solving the NF placement and flow routing problem.

B. Our Contributions

Our first contribution is that we model the joint NF placement *and* flow routing problem as a multi-objective mixed integer linear programming (MILP) problem. *The model is able to allocate NF instances and find end-to-end routes of flows while maintaining the precedence constraint among NFs of a service chain*. The MILP is solved using conventional (CPLEX) solvers for a reasonable scale problem with realistic parameters. The results provide us valuable insights to develop a heuristic solution capable of solving the NFV-RA problem for larger scale in a reasonable time.

Secondly, we develop a heuristic-based NFV-RA scheme, ClusPR, which strikes a balance between minimizing the path stretch experienced by flows and maximizing the utilization of the network. ClusPR captures the dependency between the routing and placement decisions. Both our MILP model and ClusPR consider link and node capacity constraints in making their decision. Unlike most papers in the literature that restrict the placement of NFs in the Cloud [10], [13], *ClusPR can be used in a more general setting where NFs may be hosted not only in the cloud but also on the edge computing nodes.*

To the best of our knowledge, there is no existing work that is scalable, creates a balance between minimization of the path stretch and maximization of the utilization of the network and considers the various resource limitations as our proposed approach ClusPR.

In summary, the contributions of this paper are:

- A novel multi-objective MILP formulation for the joint NF placement and flow routing problem.
- ClusPR, a heuristic-based scalable resource allocation scheme that is developed taking as input the insights observed from the MILP's results.
- Results from a number of experiments with realistic topologies demonstrate the effectiveness of ClusPR.

II. The System Model

We consider a network of nodes and links, modeled as a directed graph, $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes in the network and $\mathcal{L}$ is the set of links interconnecting the nodes. A node can be, a data-center, a router or a commercial off-the-shelf (COTS) hardware together with a router. The network carries a set of flows, $\mathcal{F}$, and supports a set of NFs, denoted as $\mathcal{V}$. For each NF type, multiple instances may be instantiated on one or multiple nodes.

A node n is characterized by the number of cores at the node, denoted as K_n. *An NF instance can be hosted on any node that has enough number of cores.* An NF instance $v \in \mathcal{V}$ hosted on node n is characterized by its service rate of requests, μ_n^v. Here, we assume that the instances of the same NF type at the same node have the same processing capacities. Note that an instance of type v may need multiple cores, k^v. In addition, an NF instance can process multiple flows whose NF service chains include the NF of the instance. We use D_n^v to denote the expected nodal delay for type v NF at node n, consisting of both processing delay and queueing delay for flows with NF type v at the node.

Each link $l \in \mathcal{L}$ is assumed to be bi-directional, and we use $l_{n'}^{n}$ to represent the link from node n to n' and C_l its expected transmission rate of bits. Each node n has a set of outgoing links represented by $\mathcal{L}_n^{out}$ and a set of incoming links, $\mathcal{L}_n^{in}$. The expected delay on link l, that comprises transmission and propagation delays, is written as D_l.

A flow $f \in \mathcal{F}$ is a sequence of data packets that are generated at expected rate λ_f and sent from a source to a destination node, traversing a sequence of intermediate nodes and links in the network. Each flow f has a specified service chain of NFs, denoted as $\overrightarrow{S}_f = (S_f^1, S_f^2 ..., S_f^{J_f})$, which is an *ordered* sequence of required NFs that the flow's packets must go through, where $S_f^j \in \mathcal{V}$ denotes the j^{th} NF on flow f's service chain and $J_f := |\overrightarrow{S}_f|$ is the length of the NF chain of flow f. We assume that the NFs in a flow's service chain are different.

In this paper, in addition to the sequence of NFs to be followed for the service chain, each flow f also has an end-to-end delay requirement, denoted as D_f, between the source node s_f and the destination node d_f of the flow. The end-to-end delay is composed of two types of delays: total delay on links, denoted as D_T and total delay on nodes, denoted as D_P.

III. The NFV Resource Allocation Problem

As discussed earlier, in a network supporting NFV, resource allocation decisions should be made both at the network-level and at the flow-level. For the former, an NFV resource allocation mechanism needs to decide the number of NF instances to instantiate in the network to process the flows and at which nodes in the network such NF instances should be placed. For the latter, the mechanism needs to decide how to route the flows to go through the NF instances according to the order of NFs in their service chains and at the same time to meet the flows' throughput and delay requirements.

A flow is admitted to the network if and only if there are NF instances that can serve all the services in the NF service chain of the flow without violating the flow's delay requirement. Indicator variable $I_n^v(f;j) = 1$ denotes that an NF instance v hosted at node n is used by the j^{th} service on the service chain of flow f, and indicator variable $I_l(f;j,n_j;j+1,n_{j+1}) = 1$ denotes that link l is used by flow f to route from the j^{th} to $(j+1)^{th}$ NF service, hosted at node n_j and n_{j+1} respectively. Since more than one instance of the same NF type may be hosted at the same node, we use an integer decision variable y_n^v to represent the number of NF type v instances that are hosted at node n. Note that, while y_n^v is a network-level decision variable that decides the number and placement of NF instances, $I_n^v(f,j)$ and $I_l(f;j,n_j,j+1,n_{j+1})$ are flow-level decision variables that specify which NF instances will be used by a given flow and which links will be used to route the flow respectively.

For this NFV resource allocation problem, three objectives are of interest: *(1) to maximize the number of flows admitted to the network, (2) to minimize the use of nodal processing capacities or cores, and (3) to minimize the utilization of link capacities*, where (2) and (3) are purposed to maximally leave resources for future use. The objective functions can then be represented as:

$$\text{max.} \quad \sum_{\forall f \in \mathcal{F}} \left\lfloor \frac{\sum_{j=1}^{J_f} \sum_{\forall n \forall v} I_n^v(f;j)}{J_f} \right\rfloor \tag{1}$$

$$\text{min.} \quad \sum_{\forall n \forall v} \frac{y_n^v k^v}{K_n} \tag{2}$$

$$\text{min.} \quad \sum_{\forall l \forall f \forall n_j \forall n_{j+1}} \sum_{\forall j}^{J_f} \frac{I_l(f;j,n_j;j+1,n_{j+1})\lambda_f}{C_l} \tag{3}$$

The above three objective functions are combined into a single-objective function, using the traditional weighted sum method [14]. Since, maximizing a given function is equivalent to minimizing the negative of the function, the single-objective function is to minimize the summation of the objective functions (2), (3) and negative of (1). The three objective functions are weighted equally, with unit weights. For positive weights, the optimal solution of the single-objective representation is also a Pareto optimal solution of the multi-objective problem [14].

A. Constraints

In solving the MILP problem, a number of constraints must be satisfied, which can be classified as: *capacity constraints, delay constraints,* and *NF chaining constraints*.

The capacity constraints are to ensure that the total traffic rate on any link does not exceed the link's transmission capacity (i.e., Constraint (III-A)), the total number of cores allocated to NF instances at any node does not exceed the cores at that node (i.e., Constraint (5)), and the total processing capacity required for admitted flows for any NF instance does not exceed that instance's processing capacity (i.e., Constraint (6)):

$$\sum_{\forall f \forall n_j \forall n_{j+1}} \sum_{\forall j}^{J_f} I_l(f;j,n_j;j+1,n_{j+1})\lambda_f \le C_l \quad \forall l \in \mathcal{L} \tag{4}$$

$$\sum_{\forall v} y_n^v k^v \le K_n \quad \forall n \in \mathcal{N} \tag{5}$$

$$\sum_{\forall f} \sum_{\forall j}^{J_f} I_n^v(f;j)\lambda_f \le \mu_n^v \quad \forall v \in \mathcal{V}, \forall n \in \mathcal{N} \tag{6}$$

Related also are two constraints implied by the definition of $I_n^v(f;j)$. One is that a flow is assigned to use NF instance of type v on node n only if there is at least one instance of NF type v hosted on node n. (i.e., Constraint (7)). Another is the constraint that any NF in the service chain of any flow is served at most by one such NF instance (i.e., Constraint (8)).

$$y_n^v \ge I_n^v(f;j) \quad \forall n \in \mathcal{N}, \forall v \in \mathcal{V}, \forall f \in \mathcal{F}, \forall j \in J_f \tag{7}$$

$$\sum_{\forall n \forall v} I_n^v(f;j) \le 1 \quad \forall f \in \mathcal{F}, \forall j \in J_f \tag{8}$$

The second category are delay constraints to ensure that a flow is admitted only if its end-to-end delay requirement is met (i.e., Constraint (3)):

$$D_T + D_P < D_f \quad \forall f \in \mathcal{F} \tag{9}$$

where $D_T = \sum_{\forall l \forall n_j \forall n_{j+1}} \sum_{\forall j}^{J_f} I_l(f;j,n_j;j+1,n_{j+1})D_l$ and $D_P = \sum_{\forall j}^{J_f} \sum_{\forall v} \sum_{\forall n} I_n^v(f;j)D_n^v$ are respectively the total link delay and the total nodal delay that the admitted flow f experiences in the network.

The third category is NF chaining constraints, which are used to ensure that the order of NFs of any flow is followed when it is routed from its source node, through NF instances of its service chain and finally to its destination node. As shown below, several constraints, i.e., (10) – (16), are involved in this category. In these constraints, variable j is used to represent the service order, i.e., $j = 1$ represents the first service, $j = 2$ the second service, and so on in an NF service chain. For the source and the destination we use $j = 0$ and $j = J + 1$, respectively.

We start with Constraints (10) and (11) that can be regarded as the flow conservation equations for the set of nodes that host NF instances assigned to serve a flow. Specifically, Constraint (10) ensures that one of the *outgoing links of node* n_j that is running the j^{th} service of flow f has to be assigned for routing flow f from its j^{th} to $(j+1)^{th}$ service order. The placement decision variables $I^v_{n_j}(f;j)$ and $I^v_{n_{j+1}}(f;j+1)$ are multiplied to make sure that the constraint applies to the cases where node n_j is used for the j^{th} service and node n_{j+1} is utilized for the $(j+1)^{th}$ service. Similarly, Constraint (11) makes sure that one of the *incoming links of node* n_{j+1} that runs $(j+1)^{th}$ service of flow f has to be assigned for routing flow f from its j^{th} to $(j+1)^{th}$ service. We remark that both (10) and (11) are not linear but since they are multiples of binary variables they can easily be substituted by a set of linear equations.

$$\forall f \in \mathcal{F}, \forall j \in 1,..J-1, \forall n_j, n_{j+1} \in \mathcal{N}:$$

$$\sum_{l \in \mathcal{L}^{out}_{n_j}} I_l(f;j,n_j;j+1,n_{j+}) I^v_{n_j}(f;j) I^v_{n_{j+1}}(f;j+) = 1 \tag{10}$$

$$\sum_{l \in \mathcal{L}^{in}_{n_{j+}}} I_l(f;j,n_j;j+1,n_{j+}) I^v_{n_j}(f;j) I^v_{n_{j+1}}(f;j+) = 1 \tag{11}$$

Constraint (12) is a flow conservation constraint of the *intermediate nodes* that do not host NF instances for the flow but still should route the flow in a given service order path. It makes sure that, if one of the incoming links of a node is assigned for a given service order then one of the outgoing links of the same node has to be assigned to the same service order.

$$\forall f \in \mathcal{F}, \forall j \in 0,..J, \forall n \in \mathcal{N}/n \neq n_j, n_{j+1}:$$

$$\sum_{l \in \mathcal{L}^{in}_{n}} I_l(f;j,n_j;j+1,n_{j+1}) - \sum_{l \in \mathcal{L}^{out}_{n}} I_l(f;j,n_j;j+1,n_{j+1} = 0, \tag{12}$$

Constraints (13) and (14) are source node flow conservation constraints. Constraint (13) ensures that one of the outgoing links of the *source node* of flow f is used to route from the 0^{th} to 1^{st} service of the service chain. As defined earlier, the source node represents the 0^{th} service. That is, node n_0 is indeed s_f, the source node of flow f. In addition, Constraint (14) is used to assign one of the incoming links of a node (n_1) serving the 1^{st} service of the service chain.

$$n_0 = s_f, \forall n_0, n_1 \in \mathcal{N}, \forall f \in \mathcal{F}:$$

$$\sum_{l \in \mathcal{L}^{out}_{n_0}} I_l(f;0,n_0;1,n_1) = I^v_{n_1}(f;1) \tag{13}$$

$$\sum_{l \in \mathcal{L}^{in}_{n_1}} I_l(f;0,n_0;1,n_1) = I^v_{n_1}(f;1) \tag{14}$$

Constraints (15) and (16) are flow conservation constraints for the destination node. Constraint (15) makes sure that one of the incoming links of the *destination node* is assigned to route from the node hosting the last NF of the service chain (n_J) to the destination node (d_f) that is also represented as the $J+1$ service, as defined earlier. Constraint (16) completes the route by assigning one of the incoming links of the destination node to the last J to $(J+1)^{th}$ service order path.

$$n_{J+1} = d_f, J = J_f, \forall n_J, n_{J+1} \in \mathcal{N}, \forall f \in \mathcal{F}:$$

$$\sum_{l \in \mathcal{L}^{in}_{n_{J+1}}} I_l(f;J,n_J;J+1,n_{J+1}) = I^v_{n_J}(f,;J) \tag{15}$$

$$\sum_{l \in \mathcal{L}^{out}_{n_J}} I_l(f;J,n_J;J+1,n_{J+1}) = I^v_{n_J}(f;J) \tag{16}$$

B. Observations from Solving the MILP Problem

We implemented our MILP model and used CPLEX to arrive at the optimal placement on a small topology of 11 nodes and 13 links. Though the results from such small scale experiments are not generalizable, the results provide us guidance to develop a heuristic solution that can solve the placement and routing problem at much larger scale. Based on solving the MILP for a number of scenarios (service chains varying in length, nodes having varying number of cores, and NFs needing multiple, but different numbers of cores), whose details are omitted due to space limitation, we make the following observations.

1) **Observation 1 (O1)**: It is desirable that all the nodes chosen for NF placement are on the shortest path of at least one of the flows. If a node that is on the shortest path of many flows has enough capacity to host all the NFs required for those flows, the optimal solution places all the NFs on that node.
2) **Observation 2 (O2)**: Different instances of the same type of NF are mostly placed on different nodes in the optimal solutions.
3) **Observation 3 (O3)**: The total number of flows that can be accommodated is a function of many factors such as the the node and link capacities, NFs' service rate and flow types. Prioritizing flows that require popular services (at least one) increases the total number of flows that can be accommodated in the network.
4) **Observation 4 (O4)**: More than the minimum number of NF instances needed might be instantiated to satisfy the performance requirements (e.g., stringent delay) of some flows.

IV. ClusPR: A Heuristic Approach for NFV Resource Allocation

Based on the observations summarized in Sec. III-B, we develop ClusPR, a heuristic-based NFV resource allocation algorithm. ClusPR consists of three phases: Initialization/**Clus**tering, **P**lacement, and **R**outing phase. Fig. 1 shows the overall design of ClusPR.

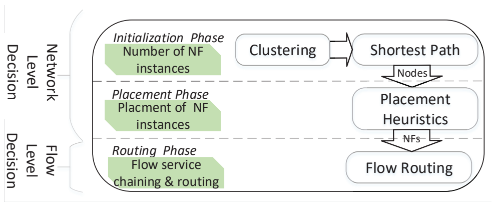


Fig. 1: ClusPR Resource Allocation Phases

Given that a number of flows potentially need to share the same NF instance, the NF may not be on the shortest path of all of the flows. Thus, a significant number of flows may have to deviate from their shortest path to be served by the NFs instantiated. However, this deviation needs to be constrained, so as not to violate the delay requirement of flows.

ClusPR uses clustering of access nodes (e.g. last hop routers to sources and destinations in a wide-area network) to minimize the deviation. Access nodes are clustered based on their proximity to each other i.e access nodes that are close to each other are clustered together. *Because of the proximity between clustered access nodes, flows originating from access nodes of a cluster and going to access nodes in another cluster have a high probability of sharing path, thus, can also potentially share NF instances with no or minimal path stretch.* Inspired by this insight, *ClusPR groups flows into intra-cluster(cluster) and inter-cluster(cluster-pair) flows and performs the placement of NF instances for each of these groups of flows independently.*

After clustering, nodes that are on the shortest path of flows are identified for each of the intra-cluster and inter-cluster set of flows. Note that the clustering is targeted at finding close by access nodes, and the shortest path nodes need not necessarily belong to a cluster or cluster-pair. The shortest path nodes that have a processing power for hosting NFs are regarded as the "best" candidates for hosting NFs. *Information about the path of each group of flows is captured through the shortest path nodes selected.* In the placement phase, *NF instances are placed on the "best" candidate shortest path nodes or their neighboring nodes.* Subsequently, flows are mapped to instances and routed while respecting their precedence constraints for service chains. A flow will be assigned to NF instances that have the minimum overall end-to-end communication cost.

A. *Initialization Phase*

The *first module* in the initialization phase is *clustering*. In this phase, access nodes are clustered based on proximity. Nodes that are either the source or destination of one or more flows are identified as access nodes. A network clustering algorithm, Kruskals algorithm [15], which is a minimum spanning tree (MST) based clustering algorithm [16], is used.

The *second module* is the *shortest path*. Dikjstra's shortest path algorithm is adapted for shortest path calculation between all pairs of access nodes (in a cluster or cluster-pair) and used to identify nodes that are on the shortest path of flows. This set of shortest path nodes are considered the "best" nodes for placing NFs (as noted in **O1**). The number and type of services required by each of the flows between access nodes can be different. To account for these differences, each node keeps a *weight* and a *list*. The *weight* is used to record the number of flows between the access nodes whose shortest path passes through the node. The *list* is used to record the different types of services these flows require.

For example, if a node is on the shortest path between three pairs of access nodes that have 3, 5 and 10 flows between them, the node will have a *weight* equal to 18. In addition, if these flows require DPI, proxy and firewall services, the node will have a *list* containing these three services.

Shortest path nodes are ordered based on their *weight:* the higher the *weight* of a node the higher its priority for hosting NF instances. In other words, nodes that are on the shortest path of many flows are given higher priority, as noted in (**O1**). If the *weight* of the nodes is equal, then nodes that have higher processing power are given priority over nodes that have lower processing power. Next the placement decisions are made for intra-cluster group of flows followed by inter-cluster flows starting from a cluster that has the largest number of intra-cluster flows.

B. *Placement Phase*

The placement phase receives the set of best candidate nodes from the initialization phase. The required type and number of instances to serve a set of flows in a cluster or cluster-pair are calculated. This set of *NF types are ordered according to their popularity, which is measured by the number of flows that require the NF.* The most popular NFs are prioritized to be placed first, considering (**O3**), with ties broken by prioritizing the NF requiring more processing power. NFs are prioritized based on their type. The number of each type of NFs to be instantiated is recorded.

Bin-Packing: The placement *heuristic does a bin-packing of the prioritized NFs on the set of "best" candidate shortest path nodes or their neighboring nodes.* An NF instance is placed on a node if and only if the node has the NF type in its *list*. This constraint is used to make sure that an NF instance placed on a node is needed by the flows whose shortest path passes through the node. An NF type that is on top of the priority queue of NF types is taken and the ordered queue of "best" candidate nodes is iterated through until a node that has the NF type in its *list* is found. Once a node is found the NF is placed and the number of instances of the NF type is decreased. The node will then be regarded as an *active node* for placing the next NF type.

Diversity: The placement *heuristic diversifies the types of NFs placed on a node*. That is, if more than one instance of a given type of NF is needed, *the algorithm prioritizes placing different types of NF instances in one node* rather than placing multiple instances of the same type of NF on the node. If different types of NFs are placed on one node, the probability that a flow can get all of the services it requires from one node will be high, as noted in (**O2**). Serving a flow's chain in one node has advantages such as decreasing the communication cost and the delay experienced by the flow.

Next, the following NF type is picked from the queue of NFs and placed on the *active node* provided that the NF is found in its *list*. If not the algorithm returns to the queue of the nodes, and looks for another node that has the NF in its *list*. After placing one instance of all types of NFs, the algorithm returns back to the top of the queue of NFs and place the second instances.This process is repeated until all the instances of all NF types are placed.

C. Routing Phase

Finally, the routing phase assigns NF instances and determines the routes for all flows. A flow will use NF instances that are placed on the set of shortest path nodes and/or their neighboring nodes for the cluster or cluster-pair the flow belongs to. Out of this set of NF instances, *a flow is assigned to NF instances that have the smallest end-to-end cost*. In order to select these NFs, *a flow's routing is modeled as a multi-stage shortest path problem* in which the stages of the multi-stage graph represent the services in the service chain of the flow.

For constructing the graph, the costs on the links of the graph need to be calculated. The costs can be calculated using shortest path algorithms such as Dijkstra's algorithm. The shortest path costs of the links from the source node to the nodes hosting the first NF instance of the chain and the links from the destination node to nodes hosting the last NF of the service chain need to be calculated for each of the flows. The costs of the links between the stages (nodes hosting NFs in the chain) are calculated only once, which decreases the computation complexity of constructing the multi-stage graph.

Dynamic programming is used to find the optimal shortest path in a multi-stage graph. To formulate the dynamic program, two distance notations are adapted. $C(n, n^{'})$ is used to represent the cost of the shortest distance between node n and $n^{'}$ that belong to two consecutive stages. e.g., $C(s_f, n_{1^{st}nf})$ represents the cost of the shortest distance between the source node of flow f (s_f) and node $(n_{1^{st}nf})$ that hosts the 1^{st} service instance. $D(n, j)$ represents the shortest distance between node n that is hosting the j^{th} service of the flow to the destination node(d_f), e.g., $D(n_{2^{nd}nf}, 2)$ represents the shortest distance from node $n_{2^{nd}nf}$ that is hosting the 2^{nd} service to the destination node.

The dynamic program formulation is given as

$$D(s_f, d_f) = \min_{n_{1^{st}} \in \mathcal{N}_{1^{th}nf}} (C(s_f, n_{1^{st}nf}) + D(n_{1^{st}nf}, 1)) \tag{17}$$

$$D(n_{j^{th}nf}, j) = \min_{n_{j^{th}nf} \in \mathcal{N}_{j^{th}nf}} (C(n_{j^{th}nf}, n_{(j+1)^{th}nf}) + D(n_{(j+1)^{th}nf}, j+1)) \tag{18}$$

$\mathcal{N}_{j^{th}nf}$ is the set of nodes that are on the shortest path or the fewest hops away from the shortest path nodes and are hosting the flow's j^{th} service type for the cluster or cluster-pair the flow belongs to. The dynamic program is solved starting from the destination node until the source node is reached. Note that the *clustering helps in decreasing the computation complexity by decreasing the possible set of instances($\mathcal{N}_{j^{th}nf}$) that a flow can choose from*, which increases the scalability of ClusPR. The algorithm checks for all the possible pairs of combinations of available instances and *chooses the instances that give the minimum overall communication cost*.

V. Experimental Results

We extensively analyze the performance of ClusPR and compare its performance with two alternatives, E2 [10] and [9] (refered to as "Deploying" for ease of reference). We report here on experiments with the Rocketfuel topology AS 1221 [17] shown in Fig. 2 used as a test network. We classify nodes in the topology as "access" (in blue), "edge" (green) and "core" (orange) nodes, in a manner similar to [18]. NFs are considered to be hosted on (or adjacent to) edge and core nodes. Each node hosting NFs has 4 cores and each instance of an NF requires one core, with a service rate of 10Mbps.There are 5 types of NFs (e.g., Firewall, Deep Packet Inspection (DPI), Network Address Translator (NAT), Intrusion Detection System (IDS) and Proxy). All the links have a capacity of 1 Gbps, and the delays on the links are: access-edge: 3 ms; edge-core: 10 ms; core-core: 40 ms. The source and destination nodes of flows as well as the services required by the flows are generated randomly. The arrival rate of flows is assumed to follow a log-normal distribution [19] and the length of the service chain for each flow is assumed to be equal to two. The service types in the chain of each flow are generated randomly. The MILP model in Section III which is solved using CPLEX is not able to scale to the network scenario considered in this experiment. The main performance measures we evaluate are total delay, path stretch and the number of flows admitted. In addition, we have done a trade-off analysis between the number of NF instances instantiated and the delay of flows.

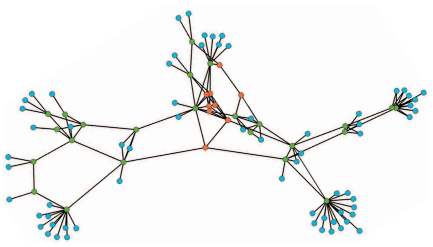
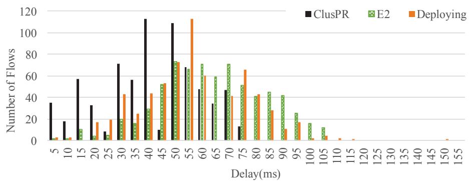
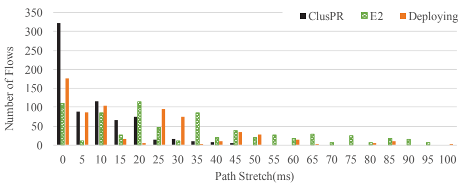


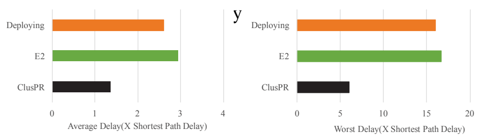
Fig. 2: Test Network:100 nodes and 294 links



(a) Distribution of Delay



(b) Additional Delay from Path Stretch



(a) Average Delay (b) Worst Delay

Fig. 4: Average and Worst-case Normalized Delay

Average and Worst-case Delays: ClusPR restricts flows to use instances placed near/on the shortest path nodes of a cluster or cluster-pair that flows belong to. This constraint helps ClusPR minimize the worst-case delay of flows. Fig. 4a and 4b show the average and worst-case normalized delays of flows respectively. Fig. 5 shows Cumulative Distribution Function(CDF) of the delay normalized with respect to the shortest path delay. ClusPR is able to achieve a worst-case normalized delay that is $9-10\times$ less than the worst-case delay of E2 and Deploying. In addition, the averaged normalized delay of ClusPR is $1.2-1.6\times$ less compared to E2 and Deploying

Total Delay and Path Stretch: The total communication delay flows experience is a summation of links propagation delay and queuing delay on the NF instances of service chains. Assuming that the queuing delay of flows is small compared to the propagation delay [20], the total delay of a flow is then measured as the summation of propagation delay on its links. The path stretch is measured as the difference between the total delay and the shortest path delay. Fig. 3 shows the total delay and path stretch distributions with 720 flows that have average flow arrival rate of 0.5 Mbps. Both E2 and ClusPR instantiated 74 instances for serving the flows.

ClusPR has a shorter path stretch compared to both Deploying and E2 for the *same number of instances*. The performance gain is achieved because ClusPR takes as input the path of flows in making NF placement decisions. ClusPR seeks to minimize the path stretch by placing NF instances near the shortest path of flows. In addition to placing NFs near the path of the flows that need them, ClusPR diversifies the type of NFs placed on a node by placing different types of NFs. This increases the probability that a flow can get all of the services of its chain in one node. Which in turn can decrease the additional path stretch incurred for fulfilling the precedence constraint. Both E2 and Deploying do not explicitly reduce the path stretch of flows.

Delay Requirement Satisfaction of Flows: For this experiment, flows are assumed to have a specified delay requirement in terms of the maximum normalized delay that they can tolerate. The normalized delay requirement of flows is assumed to be uniformly distributed between $1 - 2.5\times$ their shortest path delay.

The number of flows whose delay requirement is satisfied is shown in Fig. 6. ClusPR is able to satisfy the delay requirement of 87% of the flows while E2 and Deploying managed to fulfill the delay requirement of 60% and 70% of the flows respectively. ClusPR satisfies the delay requirement of 17% to 27% more flows compared to E2 and Deploying.

Trade-off Analysis: One of the observation from the optimal solutions of the MILP model in Section III-B(**O4**) is that more NF instances might need to be instantiated if stringent delay requirement of flows is to be satisfied. An experiment is conducted in order to analyze the trade-off between the number of NF instances instantiated and the delay performance of flows. Fig. 7 shows the path stretch of flows when ClusPR is instantiating 74 and 88 NF instances. The path stretch has decrease with an increase in the number of NF instances.

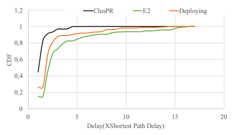


Fig. 5: CDF of Normalized Delay

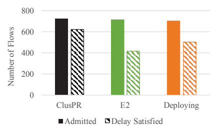


Fig. 6: Flow Statistics

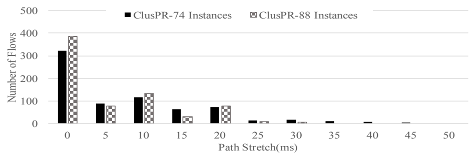
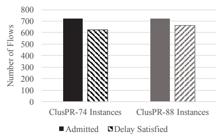
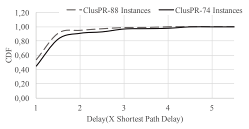


Fig. 7: Additional Delay from Path Stretch



(a) Flow Statistics



(b) CDF of Normalized Delay

Fig. 8: Trade-off Analysis B/n Number of Instances and Delay

With 74 instances the delay requirement of 87% of the flows is satisfied. By increasing 14 more NF instances, the delay requirement of 92% of the flows is satisfied as shown in Fig. 8a. As can be inferred from the CDF in Fig. 8b, the average and worst-case normalized delays have also seen a slight decrease with an increase in the number of NF instances. The average normalized delay has reduced from $1.36\times$ to $1.32\times$ and the worst-case delay has decreased from $6\times$ to $5\times$ the short path delay.

VI. Conclusion

The flexibility brought about by NFV can potentially change the way networks are managed and services are provisioned. Nevertheless, an efficient resource allocation algorithm is needed to instantiate NF instances when and where needed, and route flows through them accordingly. In this paper, a comprehensive novel multi-objective MILP model is formulated for the NFV-RA problem. Based on the useful insights obtained from the optimal solutions of the MILP model, a clustering-based heuristic scheme, ClusPR, is developed. ClusPR is scalable and balances between minimizing the path stretch and maximizing the network utilization. By factoring in information about the path of flows in the NF placement decision making and diversifying NFs, ClusPR is able to considerably minimize the path stretch. Compared to the state-of-the-art approaches ClusPR has managed to decrease the average normalized delay by a factor of $1.2 - 1.6\times$ and the worst-case delay by $9 - 10\times$, while utilizing the same number of instances. This minimization of the path stretch has enabled ClusPR to satisfy the delay requirement of 17% to 27% more flows. In addition, the trade-off between the number of NF instances and delay is shown by demonstrating the gain in delay performance for an increase in the number of NF instances.

References

[1] J. Sherry et al., "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] W. Zhang et al., "Opennetvm: A platform for high performance network service chains," in *workshop on Hot topics in Middleboxes and Network Function Virtualization*. ACM, 2016.

[3] Y. Zhang et al., "Steering: A software-defined networking for inline service chaining," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. IEEE, 2013, pp. 1–10.

[4] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 2015, pp. 50–56.

[6] R. Cohen et al., "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[7] V. Sekar et al., "Design and implementation of a consolidated middlebox architecture," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012, pp. 323–336.

[8] Q. Li, Y. Jiang, P. Duan, M. Xu, and X. Xiao, "Quokka: Latency-aware middlebox scheduling with dynamic resource allocation," *Journal of Network and Computer Applications*, vol. 78, pp. 253–266, 2017.

[9] T.-W. Kuo et al., "Deploying chains of virtual network functions: On the relation between link and server usage," in *INFOCOM, 2016 Proceedings IEEE*. IEEE, 2016.

[10] S. Palkar et al., "E2: a framework for nfv applications," in *25th Symposium on Operating Systems Principles*. ACM, 2015.

[11] B. Addis et al., "Virtual network functions placement and routing optimization," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 171–177.

[12] S. Mehraghdam et al., "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 7–13.

[13] M. Xia et al., "Network function placement for nfv chaining in packet/optical data centers," in *Optical Communication (ECOC), 2014 European Conference on*. IEEE, 2014, pp. 1–3.

[14] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[15] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[16] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 68–86, 1971.

[17] N. Spring et al., "Measuring isp topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[18] A. Afanasyev et al., "Interest flooding attack and countermeasures in named data networking," in *IFIP Networking Conference, 2013*. IEEE, 2013, pp. 1–9.

[19] Y. Zhang et al., "On the characteristics and origins of internet flow rates," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 309–322.

[20] F. P. Kelly, "Models for a self–managed internet," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 358, no. 1773, pp. 2335–2348, 2000.

Paper B

ClusPR: Balancing Multiple Objectives at Scale for NFV resource allocation

Yordanos T. Woldeyohannes, Ali Mohammadkhan, K. K. Ramakrishnan, and Yuming Jiang

In IEEE Transactions on Network and Service Management 15, no. 4 (2018): 1307-1321.

ClusPR: Balancing Multiple Objectives at Scale for NFV Resource Allocation

Y. T. Woldeyohannes, Ali Mohammadkhan, K. K. Ramakrishnan *Fellow, IEEE,* and Yuming Jiang, *Senior Member, IEEE*

***Abstract*—Network Function Virtualization (NFV) implements network middleboxes in software, enabling them to be more flexible and dynamic. NFV resource allocation methods can exploit the capabilities of virtualization to dynamically instantiate network functions (NFs) to adapt to traffic demand and network conditions. Deploying NFs requires decisions for NF placement, and routing of flows through these NFs in accordance with the sequence of NFs required to process each flow. The challenges in developing an NFV resource allocation scheme include the need to manage the dependency between flow-level (routing) and network-level (placement) decisions and to efficiently utilize resources that may be distributed network-wide, while fulfilling the performance requirements of flows.**

We propose a scalable resource allocation scheme, called ClusPR, that addresses these challenges. By elegantly capturing the dependency between flow routing and NF placement, ClusPR strikes a balance between multiple objectives including *minimizing path stretch*, *balancing the load among NF instances*, while *maximizing the total network utilization* by accommodating the maximum number of flows possible. ClusPR addresses the offline problem of NFV resource allocation. To address the online problem of dynamically placing and routing flows upon their arrival, we propose iClusPR. iClusPR is an online algorithm that performs dynamic scaling by adjusting the number of NF instances based on the traffic demand and the network state.

Our experiments show that ClusPR achieves the near-optimal solution for a practical large-sized network in reasonable time. Compared to the state-of-the-art approaches, ClusPR decreases the average normalized delay by a factor of $1.2 - 1.6\times$ and the worst-case delay by more than $10\times$, with the same or slightly better network utilization and balances the load among NF instances. Furthermore, the performance of iClusPR, the online version, is comparable to the offline ClusPR algorithm.

***Index Terms*—NFV, orchestration, placement, flow routing, multi-objective, load balancing, clustering.**

I. Introduction

Middleboxes such as firewalls, VPN gateways, proxies, intrusion detection and prevention systems play a central role in today's Internet by providing network resident functionality that examines and potentially modifies the end-to-end traffic flow [2]. Implementing middleboxes is gradually migrating to software platforms, providing additional flexibility and extensibility for the capabilities of the network compared to purpose-built hardware appliances. Evolving the network's capabilities can thus involve lower capital expenditures as the software can run on commercial off-the-shelf (COTS) hardware. Network Function Virtualization (NFV) decouples the software of network functions from the physical machine and runs it on virtual machines, or more recently on "containers" [3]. This also brings greater flexibility in resource management as instances of the network functions (NFs) can be created dynamically, and the capacity for a particular function can be scaled up or down depending on the traffic demand. Sequences of NFs are common, and the overall service provided by the network by such sequences of NFs is termed "service function chaining" [4]. Using the capability of Software Defined Networks (SDN) to perform flow specific routing, we can route the flows requiring the service chain functionality through the NF service instances.

A. The Challenge

Resource allocation of NF service chains continues to be a challenge because of the complexity of making network-level as well as flow-level decisions holistically, since each impacts the other. An NFV resource allocation (NFV-RA) mechanism has determine at the network-level the number of NF instances to instantiate across all the nodes in the network to process all admitted flows. Further, the algorithm needs to determine the placement of the NFs or the physical machines that should host the NFs. The flow-level decision making includes the assignment of NF instances for the service chain of the flow and the determination of the route for each flow. However, the network-level decision making has to be coupled with the decision making at the flow-level, because placement decisions resulting in NFs placed on network nodes not along the shortest path will result in "path stretch", contributing to increased latency for the flow. This interdependence between network-level (placement, instantiating the requisite number of NFs) and flow-level (instance assignment, routing) decisions makes the NFV-RA problem new and challenging, warranting the recent attention it has received in the research community.

In addition, network resources such as bandwidth of links, memory and processing capacities of nodes are limited. These resource limitations call for efficient utilization of the available resources, which could be achieved for example by minimizing the number of NF instances, balancing the load among the NF instances and/or minimizing the number of distinct resources used. However, these efficient resource utilization objectives could sometimes be in conflict with the objective of fulfilling the performance requirement of flows. Thus, one of the challenges in developing an NFV resource allocation scheme is to

Y. T. Woldeyohannes and Yuming Jiang are with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology, NTNU, Norway

Ali Mohammadkhan and K. K. Ramakrishnan are with the Department of Computer Science and Engineering, University of California, Riverside, California, USA.

Preliminary version of this article appeared in ICIN, 2018 conference [1].

find a balance across multiple objectives, involving minimizing path stretch, maximizing the number of flows accommodated, and balancing the load among NF instances, while considering the resource constraints in the network.

The end-to-end latency for flows can potentially decrease if NF instances are placed closer to users instead of routing the network traffic to a limited number of centralized data-centers [5]. However, most of the work in the literature focus only on the placement of NFs in the Cloud e.g., Stratos [6] and [7] or at central offices (COs), as in E2 [8]. In this paper, we *develop a scalable resource allocation scheme for the joint NF placement and flow routing problem in a geo-distributed, general, network. The novelty of our approach comes from considering multiple objectives at the same time (minimizing path stretch, maximizing the total utilization of the network and balancing the load among NF instances).* Our work furthers the state of the art by capturing the inter-dependency between flow-level and network-level decisions in a scalable manner and striking a balance across multiple objectives. While existing works typically restrict the flow to the shortest-path avoiding path stretch, and disregarding its effect on network utilization [9], [10]. Others maximize network utilization at the expense of increased path stretch [11], [12].

B. Our Contributions

Our first contribution is that we model the joint NF placement *and* flow routing problem as a multi-objective integer linear programming (ILP) problem. The model is able to allocate NF instances and find the end-to-end route of flows while maintaining the precedence constraint among NFs of the service chain. We solve the ILP using a conventional solver (CPLEX) for a reasonable scale problem with realistic parameters. Although the ILP formulation gives optimal results, the run time of the algorithm increases exponentially with the size of the network and/or the number of flows. This is because the joint NF placement and flow routing problem is *NP-hard*, encompassing the two NP-hard problems (placement and flow routing) [13]. The results obtained from solving the ILP model provide us with valuable insights to develop a heuristic solution capable of solving the NFV-RA problem at a larger scale in reasonable time.

Secondly, taking as input the insights, we develop a heuristic-based NFV-RA scheme called ClusPR. ClusPR strikes a balance between multiple objectives including *minimizing the path stretch* experienced by flows, *maximizing the total utilization of the network* (the number of flows admitted) and *balancing the load among NF instances*. ClusPR utilizes a divide-and-conquer approach, decomposing the NFV-RA problem into two sub-problems: for NF placement and flow routing. To capture the dependency between the network-level (NF placement) and flow-level (flow routing) decisions, ClusPR adapts a novel hierarchical architecture in which first flows are grouped based on their path proximity. Then, the route information of flows is extracted and used as input when making NF placement decisions.The features of ClusPR are:

- **Network:** ClusPR can be used in general settings where NF instances may be hosted not only in the cloud but also on the edge computing nodes.
- **Path stretch vs utilization:** Path stretch can be avoided if NF instances that are needed by a flow are placed on its (shortest) path as done in [9]. This can however lead to low utilization of the network. ClusPR strikes a balance between minimizing path stretch and maximizing the network utilization.
- **Load balancing vs path stretch:** Studies have demonstrated that NF overload is a common cause of failures and therefore it is important to balance the load across NFs [2], [14]. However, such load balancing decisions might lead to increased delay and thus violating Service Level Agreement (SLA) for flows. Our proposed flow routing algorithm balances the load among NF instances while taking into account the delay requirement of flows without redirect existing flows.

Finally, we propose an online algorithm iClusPR, which dynamically scales the number of NF instances depending on the traffic demand and network state. iClusPR makes resource allocation decisions on a time slot basis.

- **Online algorithm features:** iClusPR has the same design principle as the offline algorithm, ClusPR, so it also has the aforementioned features of ClusPR.
- **Experiment:** The performance of iClusPR is analyzed through realistic experiments in which flows arrive randomly and depart after being served for a random amount of time.

II. The System Model

We consider a network of nodes and links, modeled as a directed graph, $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes in the network and $\mathcal{L}$ is the set of links interconnecting the nodes. A node can be in a data-center with multiple servers, a router or a commercial off-the-shelf (COTS) server along with a router. The network carries a set of flows, $\mathcal{F}$, and supports a set of NFs, denoted as $\mathcal{V}$. For each NF type, multiple instances may be instantiated on one or multiple nodes.

A node n is characterized by the number of CPU cores at the node, denoted as K_n, and memory capacity, denoted as M_n. A CPU core is dedicated to a single NF instance [15], and does not span CPU cores to avoid Non-Uniform Memory Access (NUMA) overheads. An NF instance can be hosted on any node that has enough available resources.

An NF instance $v \in \mathcal{V}$ hosted on node n is characterized by its service rate of requests, μ_n^v. Here, we assume that the instances of the same NF type at the same node have the same service rate. An instance of type v needs k^v cores and m^v amount of memory. In addition, an NF instance can process multiple flows whose service chains include the type of NF instance. We use D_n^v to denote the expected nodal delay for type v NF at node n, consisting of both processing delay and queuing delay for flows with NF type v at the node.

Each link $l \in \mathcal{L}$ is assumed to be bi-directional, and we use $l_{n'}^{n}$ to represent the link from node n to n' and C_l its expected transmission rate. Each node n has a set of outgoing links represented by $\mathcal{L}_n^{out}$ and a set of incoming links, $\mathcal{L}_n^{in}$. The expected delay on link l, that comprises transmission and propagation delays, is written as D_l.

A flow $f \in \mathcal{F}$ is a sequence of data packets that are generated at expected rate λ_f and sent from a source to a destination node, traversing a sequence of intermediate nodes and links in the network. Each flow f has a specified service chain of NFs, denoted as $\overrightarrow{S}_f = (S_f^1, S_f^2 ..., S_f^{J_f})$, which is an *ordered* sequence of required NFs that the flow's packets must go through, where $S_f^j \in \mathcal{V}$ denotes the j^{th} NF on flow f's service chain and $J_f := |\overrightarrow{S}_f|$ is the length of the NF chain of flow f.

In addition to the sequence of NFs to be followed for the service chain, each flow f also has an end-to-end delay requirement, denoted as D_f, between the source node s_f and the destination node d_f of the flow. The end-to-end delay is composed of two types of delays: total delay on links, denoted as D_T, and total delay on nodes, denoted as D_P.

III. The NFV Resource Allocation Problem

As discussed earlier, in a network supporting NFV, resource allocation decisions should be made both at the network-level and the flow-level. For the former, an NFV resource allocation mechanism needs to decide the number of NF instances to instantiate in the network to process the flows and where or at which nodes in the network such NF instances should be placed. For the latter, the mechanism needs to decide how to route the flows to go through the NF instances according to the order of NFs in their service chains and at the same time try to meet the flows' delay requirements by limiting path stretch.

For this NFV resource allocation problem, three objectives are of interest: *(1) to maximize the number of flows admitted to the network, (2) to minimize the use of nodal processing capacities or cores, and (3) to minimize the utilization of link capacities*, where (2) and (3) are purposed to maximally leave resources for future use.

A flow is admitted to the network if and only if there are NF instances that can serve all the services in the NF service chain of the flow without violating the flow's delay requirement. Let indicator variable $I_n^v(f;j) = 1$ denoting that an NF instance v hosted at node n is used by the j^{th} service on the service chain of flow f, and indicator variable $I_l(f;j,n_j;j+1,n_{j+1}) = 1$ denoting that link l is used by flow f to route from the j^{th} to $(j+1)^{th}$ service/NF hosted at node n_j and n_{j+1} respectively. Since more than one instance of the same NF type may be hosted at the same node, we use an integer decision variable y_n^v to represent the number of type v NF instances that are hosted at node n. Note that, while y_n^v is a network-level decision variable that decides the number and placement of NF instances, $I_n^v(f;j)$ and $I_l(f;j,n_j;j+1,n_{j+1})$ are flow-level decision variables that specify which NF instances will be used by a given flow and which links will be used in routing that flow through, respectively.

The three objectives can then be respectively represented as

$$\text{maximize} \quad \sum_{\forall f \in \mathcal{F}} \left\lfloor \frac{\sum_{j=1}^{J_f} \sum_{\forall n \forall v} I_n^v(f;j)}{J_f} \right\rfloor \tag{1}$$

$$\text{minimize} \quad \sum_{\forall n \forall v} \frac{y_n^v k^v}{K_n} \tag{2}$$

$$\text{minimize} \quad \sum_{\forall l \forall f} \sum_{\forall j}^{J_f} \frac{I_l(f;j,n_j;j+1,n_{j+1})\lambda_f}{C_l} \tag{3}$$

The above three objective functions are combined into a single-objective function, using the traditional weighted sum method [16]. Since, maximizing a given function is equivalent to minimizing the negative of the function, the single-objective function is to minimize the summation of the objective functions (2), (3) and negative of (1). The objective functions are weighted equally, with unit weights. For positive weights, the optimal solution of the single-objective representation is also a Pareto optimal solution of the multi-objective problem [16].

A. Constraints

In solving the ILP problem, a number of constraints must be satisfied, which can be classified into three categories: *capacity constraints, delay constraints,* and *NF chaining constraints*. The first category is capacity constraints, which ensure that the total traffic rate on any link does not exceed the link's transmission capacity (i.e., Constraint (4)), the total number of cores allocated to NF instances at any node does not exceed the number of cores at this node (Constraint (5)) and memory capacity (Constraint (6)), and the total processing capacity required for the admitted flows flowing through any NF instance does not exceed that instance's processing capacity (Constraint (7)):

$$\sum_{\forall f} \sum_{\forall j}^{J_f} I_l(f;j,n_j;j+1,n_{j+1})\lambda_f < C_l \quad \forall l \in \mathcal{L} \tag{4}$$

$$\sum_{\forall v} y_n^v k^v < K_n \quad \forall n \in \mathcal{N} \tag{5}$$

$$\sum_{\forall v} y_n^v m^v < M_n \quad \forall n \in \mathcal{N} \tag{6}$$

$$\sum_{\forall f} \sum_{\forall j}^{J_f} I_n^v(f;j)\lambda_f < \mu_n^v \quad \forall v \in \mathcal{V}, \forall n \in \mathcal{N} \tag{7}$$

Related and hence put in this category are two constraints implied by the definition of $I_n^v(f;j)$. One is that a flow is assigned to use NF instance of type v on node n only if there is at least one instance of NF type v hosted on node n. (i.e., Constraint (8)). Another is the constraint that any NF in the service chain of any flow is served by only one such NF instance (i.e., Constraint (9)).

$$y_n^v \geq I_n^v(f;j) \quad \forall n \in \mathcal{N}, \forall v \in \mathcal{V}, \forall f \in \mathcal{F}, \forall j \in \{1 \dots J_f\} \tag{8}$$

$$\sum_{\forall n \forall v} I_n^v(f;j) = 1 \quad \forall f \in \mathcal{F}, \forall j \in \{1 \dots J_f\} \tag{9}$$

The second category are delay constraints to ensure that a flow is admitted only if its end-to-end delay requirement is met (i.e., Constraint (10)):

$$D_T + D_P < D_f \quad \forall f \in \mathcal{F} \tag{10}$$

where $D_T = \sum_{\forall l} \sum_{\forall j \forall n_j \forall n_{j+1}}^{J_f} I_l(f; j, n_j; j+1, n_{j+1}) D_l$ and $D_P = \sum_{\forall j}^{J_f} \sum_{\forall n \forall v} I_n^v(f; j) D_n^v$ are respectively the total link delay and the total nodal delay that the flow f will experience in the network if admitted.

The third category is NF chaining constraints, which are to ensure that the order of NFs of any flow is followed when it is routed from its source node, through NF instances of its service chain and finally to its destination node. As shown below, several constraints, i.e., (11) – (20), are in this category. In these constraints, variable j is used to represent the service order, i.e., $j = 1$ represents the first service, $j = 2$ the second service, and so on in an NF service chain. The source and the destination are represented by $j = 0$ and $j = J + 1$ respectively.

We start with Constraints (11) and (12) that can be regarded as the flow conservation equations for the set of nodes that host NF instances assigned to serve a flow. Specifically, Constraint (11) ensures that one of the *outgoing links of node* n_j that is running j^{th} service of flow f has to be assigned for routing flow f from its j^{th} to $(j+1)^{th}$ service order. The NF assignment decision variables $I_{n_j}^v(f; j)$ and $I_{n_{j+1}}^v(f; j+1)$ are multiplied to make sure that the constraint applies to the case where node n_j is used to serve the j^{th} service and node n_{j+1} the $(j+1)^{th}$ service of flow f. Similarly, Constraint (12) makes sure that one of the *incoming links of node* n_{j+1} that runs $(j+1)^{th}$ service of flow f has to be assigned for routing flow f from its j^{th} to $(j+1)^{th}$ service. We remark that both (11) and (12) are not linear but since they are multiples of binary variables, they can easily be substituted by a set of linear equations.

$\forall f \in \mathcal{F}, \forall j \in \{1 \dots J_f - 1\}, \forall n_j, n_{j+1} \in \mathcal{N}$:

$$\sum_{l \in \mathcal{L}_{n_j}^{out}} I_l(f; j, n_j; j+1, n_{j+1}) I_{n_j}^v(f; j) I_{n_{j+1}}^v(f; j+1) = 1 \tag{11}$$

$$\sum_{l \in \mathcal{L}_{n_{j+1}}^{in}} I_l(f; j, n_j; j+1, n_{j+1}) I_{n_j}^v(f; j) I_{n_{j+1}}^v(f; j+1) = 1 \tag{12}$$

Constraints (13) and (14) are used to guarantee that no more than one link outgoing from or incoming to a node are assigned to a given service order of a flow respectively:

$\forall n \in \mathcal{N}, \forall f \in \mathcal{F}, \forall j \in \{0, ..J_f\}$:

$$\sum_{l \in \mathcal{L}_n^{out}} \sum_{n_j, n_{j+1} \in \mathcal{N}} I_l(f; j, n_j; j+1, n_{j+1}) \le 1 \tag{13}$$

$$\sum_{l \in \mathcal{L}_n^{in}} \sum_{n_j, n_{j+1} \in \mathcal{N}} I_l(f; j, n_j; j+1, n_{j+1}) \le 1 \tag{14}$$

Constraint (15) is a flow conservation constraint of the *intermediate nodes*, which are nodes that do not host NF instances that are assigned to the flow but still should route the flow in a given order of the route. It makes sure that, if one of the incoming links of a node is assigned for a given order of the route then one of the outgoing links of the same node has to be assigned to the same order.

$$\sum_{l \in \mathcal{L}_n^{in}} I_l(f; j, n_j; j+1, n_{j+1}) - \sum_{l \in \mathcal{L}_n^{out}} I_l(f; j, n_j; j+1, n_{j+1}) = 0, \forall f \in \mathcal{F}, \forall j \in \{0, ..J_f\}, \forall n \in \mathcal{N}/n \neq n_j, n_{j+1}, s_f, d_f \tag{15}$$

Constraints (16) and (17) are *source node* flow conservation constraints. Constraint (16) ensures that one of the outgoing links of the source node of flow f is used to route from the 0^{th} to 1^{st} service of the service chain. As defined earlier, the source node represents the 0^{th} service. That is, node n_0 is equivalent to s_f, the source node of flow f. In addition, Constraint (17) is used to assign one of the incoming links of a node (n_1), serving the 1^{st} service of the service chain, to the 1^{st} service order of the service chain.

$n_0 = s_f, \forall n_1 \in \mathcal{N}, \forall f \in \mathcal{F}$:

$$\sum_{l \in \mathcal{L}_{n_0}^{out}} I_l(f; 0, n_0; 1, n_1) = I_{n_1}^v(f; 1) \tag{16}$$

$$\sum_{l \in \mathcal{L}_{n_1}^{in}} I_l(f; 0, n_0; 1, n_1) = I_{n_1}^v(f; 1) \tag{17}$$

Constraints (18) and (19) are flow conservation constraints for the (*destination node*). Constraint (18) makes sure that one of the incoming links of the destination node is assigned to route from the node hosting the last NF of the service chain (n_J) to the destination node (d_f) that is also represented as the $J + 1$ service, as defined earlier. In addition, Constraint (19) assigns one of the outgoing links of the node serving the last service to the $(J+1)^{th}$ service order.

$n_{J+1} = d_f, \forall n_J \in \mathcal{N}, \forall f \in \mathcal{F}$:

$$\sum_{l \in \mathcal{L}_{n_{J+1}}^{in}} I_l(f; J, n_J; J+1, n_{J+1}) = I_{n_J}^v(f; J) \tag{18}$$

$$\sum_{l \in \mathcal{L}_{n_J}^{out}} I_l(f; J, n_J; J+1, n_{J+1}) = I_{n_J}^v(f; J) \tag{19}$$

Finally, Constraint (20) is used to avoid loops: If link $l_{n'}^{n}$ is assigned for a given service order then link $l_n^{n'}$ should not be assigned to the same service order.

$$I_{l_{n'}^{n}}(f; j, n_j; j+1, n_{j+1}) + I_{l_n^{n'}}(f; j, n_j; j+1, n_{j+1}) \le 1, \quad \forall l \in \mathcal{L}, \forall f \in \mathcal{F}, \forall j \in \{0 \dots J_f\}, \forall n_j, n_{j+1} \in \mathcal{N} \tag{20}$$

B. Observations from Solving the ILP Problem

We implemented our ILP model and used CPLEX to arrive at the optimal placement on a number of small size networks. We realize that the results from these small scale experiments are not generalizable, but the results provide us with valuable guidance to develop a heuristic solution that can solve the placement and routing problem at a much larger scale. Based on solving the ILP model for a number of scenarios (service chains varying in length, nodes having varying number of cores, and NFs needing multiple, but different numbers of cores), we make the following observations.

1) **Observation 1 (O1)**: The nodes chosen for NF placement are usually on the shortest path of at least one of the flows. If a node that is on the shortest path of all the flows has enough capacity to host all the NFs required for those flows, the optimal solution places all the NFs on that node.
2) **Observation 2 (O2)**: In the optimal solution, different NF types are usually placed on a node rather than multiple instances of the same NF type. If various types of NFs are placed on a node, a flow will be able to get all the services of its chain in one node with a high probability. This is especially true for flows with shorter service chains.
3) **Observation 3 (O3)**: The total number of flows that can be accommodated in the network is a function of many factors, including the amount of network resources available, the NFs' service rate and flow types. Prioritizing flows that require popular services (at least one) increases the total number of flows that can be accommodated in the network. This is because, if instances that are needed by very few flows are instantiated a priori, those instances will occupy resources but will be under utilized, thus not accommodating popular NFs that could have been utilized more.
4) **Observation 4 (O4)**: To satisfy the performance requirements (e.g., stringent delay) of some flows, more than the minimum number of NF instances needed may be instantiated.

IV. ClusPR: A Heuristic Approach for NFV Resource Allocation

Based on the observations summarized in Sec. III-B, we develop ClusPR, a scalable NFV resource allocation algorithm. ClusPR consists of three phases: Initialization/**Clus**tering, **P**lacement, and **R**outing phase. Fig. 1 shows the overall design of ClusPR.

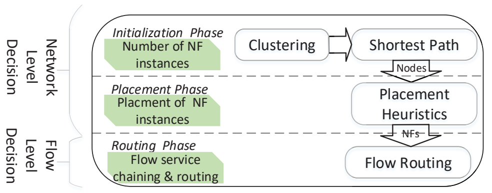


Fig. 1: ClusPR Resource Allocation Phases

Network resources (such as CPU cores, memory, bandwidth) should be used efficiently so as to maximize the total utilization of the network (i.e., the total number of flows admitted to the network with the minimum resources). On the other hand, path stretch of flows needs to be minimized, as increased latency could lead to violation of a flow's SLA. However, these objectives could be conflicting. For example, path stretch can be avoided if flows are served by NF instances placed on their shortest path as in [9], [10]. However, relatively few flows share a specific end-to-end path and those that do might have different service chain requirements. So this could lead to under-utilization of the network. For increasing the utilization of the network, an NF instance should serve a larger number of flows.

Given that a number of flows may need to share the same NF instance, that NF may not be on the shortest path of all of the flows it serves. Thus, a significant number of flows may have to deviate from their shortest path to be served by the NFs instantiated. However, this deviation needs to be constrained so as not to violate the delay requirement of flows. To strike a balance between minimizing path stretch and maximizing network utilization, ClusPR groups flows based on their path information. That is, flows whose paths are in close proximity to each other will be grouped together. Then flows within a group will share NF instances. The intuition is that since flows within a group have paths that are 'close' to each other, they can share NF instances with zero or minimal path stretch.

After grouping the flows, their path information is extracted by identifying the nodes that are on the shortest path of each group of flows. Then, NF instances are placed and assigned to flows, and flows are routed end-to-end. In Summary, ClusPR's design principle is to first *(1)group flows based on their path proximity, (2) then extract the flows' path information, (3) place instances considering the path information and (4) finally, assign NF instances and route flows* taking into account their performance requirement and the utilization level of the instances.

A. *Initialization Phase*

*1) **Clustering**:* The purpose of the clustering module is to find a set/group of flows whose paths are in close proximity to each other. To group the flows, ClusPR clusters access nodes (e.g., first/last hop routers to sources and destinations in a wide-area network). Access nodes are clustered based on their proximity to each other, i.e., access nodes that are close to each other are clustered together. *Because of the topological proximity between clustered access nodes, flows originating from access nodes of a cluster and going to access nodes in another cluster or vice versa will have their paths close to each other.* Thus, these flows can also potentially share NF instances with minimal path stretch. Inspired by this insight, ClusPR groups flows into intra-cluster (i.e., flows whose source and destination nodes are in the same access cluster) and inter-cluster flows (i.e., flows whose source and destination nodes belong to two different clusters) and performs the placement of NF instances for each of these groups of flows independently.

To cluster the access nodes, a network clustering algorithm, Kruskal's algorithm [17], which is a minimum spanning tree (MST) based clustering algorithm [18], is used. Given a network graph $G(\mathcal{N}, \mathcal{L})$, the algorithm first organizes all nodes/vertices of the graph in disjoint sets with a set containing one node/vertex. The edges of the graph are sorted and listed in ascending order, based on their weight, which is equal to the delay on the edge. Then, an edge that is on top of the sorted list of edges is considered. If the vertices of the edge belong to two different disjoint sets, the sets will be merged into a single set. In order to find an MST, this step is repeated until all the edges in the sorted list are visited.

Given the number of clusters k, an MST based clustering algorithm can be used to find the k clusters by sorting edges and merging sets until the heaviest $k-1$ edges are left in the sorted edge list [19].

*a) **Optimal Number of Clusters**:* Kruskal's algorithm can be used to find clusters given that the number of clusters is known. But finding the optimal number of clusters apriori might be challenging especially if the network size is large. To solve this problem, cluster validation techniques can be used to automatically find the optimal number of clusters [20]. One of the classical cluster validation techniques is the Dunn index [21]. The Dunn index is useful for finding dense and well-separated clusters. It is defined as the ratio between the minimum inter-cluster distance to the maximum intra-cluster distance.

$$\mathcal{D}(k) = \min_{i,j \in \{1 \ldots k\}, i \neq j} \left\{ \frac{\delta_{i,j}}{\max_{1 \leq l \leq k} \triangle_l} \right\} \tag{21}$$

Here $\delta_{i,j}$ is the inter-cluster distance between clusters i and j, defined as the minimum distance between a pair of nodes across clusters i and j, i.e., $\min_{x_i \in C_i}\{\min_{x_j \in C_j} d(x_i, x_j)\}$, C_i is the set of nodes in cluster i. The distance metric, $d(x_i, x_j)$, is the edge cost or the shortest path cost between the nodes. The intra-cluster distance of a given cluster l, $\triangle_l$, is defined as the maximum distance between a pair of nodes within that cluster i.e., $\max_{x_l \in C_l}\{\max_{x_k \in C_l} d(x_l, x_k)\}$.

The Dunn index will be maximum when the minimum inter-cluster distance is large and the maximum intra-cluster distance is small. Larger values of the Dunn index corresponds to good clusters. Therefore, *the number of clusters (k) that maximizes the Dunn index is taken as the optimal number of clusters* [22].

Algorithm 1 ClusPR: Clustering Phase

```
1:  $G(\mathcal{N}, \mathcal{L})$: where $n$ , $n'$ $\in \mathcal{N}$
2:  sort the links/edges $l \in \mathcal{L}$ in ascending order
3:  $k \leftarrow N$ number of disjoint sets(clusters)
4:  $k_{opt} \leftarrow k$
5:  for  $l : \{n, n'\} \in \mathcal{L}$ do
6:      if $n$ and $n'$ are in disjoint sets then
7:          merge sets' of $n$ and $n'$
8:          $k \leftarrow k - 1$
9:          if $\mathcal{D}(k) > \mathcal{D}(k_{opt})$ then
10:             $k_{opt} \leftarrow k$
11: return $k_{opt}$
12: Kruscal's Algorithm( $k_{opt}$)
```

Finding the Optimal Number of Clusters: To find the optimal number of clusters, Kruskal's algorithm can be run multiple times for different numbers of clusters, and the number of clusters that maximizes the Dunn index value will be considered the optimal. However, this approach can be computationally cumbersome as it requires running the clustering algorithm multiple times. To efficiently calculate the optimal number of clusters, we propose an approach that exploits the structure of Kruskal's clustering algorithm.

Kruskal's clustering algorithm is a hierarchical algorithm in which the number of clusters decreases at each step until the target number of clusters is reached. In the beginning, the number of clusters is equal to the number of nodes in the network (N). The number of clusters decreases step by step from N until the target number of clusters is reached. The Dunn index can be calculated in the middle of the MST based clustering algorithm execution.

For example, in the beginning the Dunn index can be calculated for N clusters, before two clusters are merged and then the number of clusters decreases to N-1. The Dunn index is then calculated for N-1 clusters before two sets are merged and the cluster number becomes N-2 and so on. Algorithm 1 shows the pseudo code for efficiently calculating the Dunn index inside the Kruskal's clustering algorithm. At line 10, the Dunn index for the current number of clusters k is calculated using equation (21). If it is more than the Dunn index of the temporarily optimal number of clusters (k_{opt}), k will take the place of k_{opt}. After getting the optimal number of clusters that has the maximum value of the Dunn index, Kruskal's clustering algorithm will finally be run for the optimal number of clusters (line 13).

Number of NF Instances: Latency-sensitive flows have less tolerance for path stretch. Thus, as noted in **O4**, a larger number of NF instances might have to be instantiated to satisfy the delay requirement for this type of flows. In this section, we propose two ways for deciding the number of NF instances to be created in the network. In the first approach, the minimum number of instances required to serve all the flows in the set ($\mathcal{F}$) is calculated first, then these instances are divided among the groups of flows. In the second approach, the minimum number of instances needed to serve each group of flows is calculated first, then the total number of instances instantiated will be a summation of the number of instances created for the groups.

First approach: this approach creates the minimum number of instances needed to serve all the flows in the set $\mathcal{F}$.

Theorem 1. *Given a set of flows ($\mathcal{F}$), each flow having an average arrival rate of λ, out of which F^v number of flows require the v type NF. The minimum number of NF instances of type $v \in \mathcal{V}$ ($\mathcal{I}^v_{min}$), with service rate μ^v, required to serve the set of flows is equal to $\mathcal{I}^v_{min} = \left\lceil \frac{F^v \lambda}{\mu^v} \right\rceil$.*

Proof: refer to Appendix A.

The minimum number of instances of each type of NF ($v \in \mathcal{V}$) needed to serve the flows in $\mathcal{F}$ is calculated using Theorem 1. The calculated number of instances will then be divided among the groups of flows.

- *For large numbers of flows*: If the number of flows in $\mathcal{F}$ is large, the minimum number of NFs calculated can be proportionally divided among the groups of flows. Let $\mathcal{G}$ represent the set of groups of flows. For each group $g \in \mathcal{G}$, the number of NF instances of type v instantiated ($\mathcal{I}^v_g$) is allocated in proportion to the number of flows in the group that require NF type v (F^v_g) to the total number of flows that need v, F^v. That is $\mathcal{I}^v_g = \frac{\mathcal{I}^v_{min} F^v_g}{F^v}$. Since this fraction might not always be an integer number, two conditions are used for assigning positive integer values to $\mathcal{I}^v_g$. If $\frac{\mathcal{I}^v_{min} F^v_g}{F^v} > 1$, $\mathcal{I}^v_g = \lfloor \frac{\mathcal{I}^v_{min} F^v_g}{F^v} \rfloor$. If $0 < \frac{\mathcal{I}^v_{min} F^v_g}{F^v} < 1$, $\mathcal{I}^v_g = \lceil \frac{\mathcal{I}^v_{min} F^v_g}{F^v} \rceil$. The latter condition is used to instantiate one instance for groups that have a smaller number of flows. This approach works best when the number of flows is large ($\mathcal{I}^v_{min}$ is large).

- *For small numbers of flows:* If the number of flows is small and the objective is to minimize the number of NF instances, the clustering module could be skipped. In this case, all flows will be in one group and $\mathcal{I}^v = \mathcal{I}^v_{min}$.

Second approach: In this approach the minimum number of NF instances needed to serve each group of flows ($g \in \mathcal{G}$) is calculated using Theorem 1. ClusPR groups flows into intra-cluster and inter-cluster sets of flows. If there are k access clusters, there will be $\frac{k(k-1)}{2}$ number of cluster-pairs. Thus, there will be a maximum of k groups of intra-cluster flows and $\frac{k^2-k}{2}$ groups of inter-cluster flows. The maximum number of groups of flows in $\mathcal{G}$ is equal to $\frac{k^2+k}{2}$. $\mathcal{I}^v_{g,min}$ is the minimum number of NF instances of type v needed for serving the set of flows in group g. The maximum total number of NF instances instantiated by ClusPR ($\mathcal{I}^v_{max}$) is a summation of the number of instances instantiated for each of the groups. That is, $\mathcal{I}^v_{max} = \sum_{\forall g} \mathcal{I}^v_{g,min}$.

Theorem 2. *The number of NF instances of type $v \in \mathcal{V}$ instantiated by ClusPR, $\mathcal{I}^v$, is bounded by $\mathcal{I}^v_{min} \leq \mathcal{I}^v \leq \mathcal{I}^v_{min} + \frac{k^2+k}{2}$ where k is the number of access clusters.*

Proof: refer to Appendix B.

From Theorem 2, it can be observed that the number of NF instances created by ClusPR using the second approach is upper bounded. And the upper bound is a function of the number of access clusters.

*2) **Shortest Path**:* The second module in the initialization phase is the shortest path module. The purpose of this module is to obtain path information about the groups of flows by identifying nodes that are on the shortest path of flows. The shortest path between access nodes of the flows is calculated using classical shortest path algorithms such as Dijkstra's algorithm. The nodes that are on the shortest paths are regarded as the "best" candidates for hosting NF instances (as noted in **O1**). In addition, one hop and two hops neighboring nodes of the shortest path nodes are also identified. This is done to increase the number of candidate nodes for hosting NF instances as the shortest path nodes might not have enough resources. The path information captured through the selected shortest path and neighboring nodes is then transferred to the placement phase.

Flows have various service chain requirements. To ensure that NF instances placed on a shortest path node are needed by flows whose shortest path passes through the node, each shortest path node keeps a *list*. The *list* is used to record the different types of services the flows require. In addition, the nodes will have a *weight* that is used to record the number of flows whose shortest path passes through the node. For example, if a node is on the shortest path between three pairs of access nodes that have 3, 5 and 10 flows between them, the node will have a weight equal to 18. In addition, if these flows require DPI, proxy and firewall services, the node will have a list containing these three services.

Shortest path nodes are ordered based on their weight: the higher the weight of a node the higher its priority for hosting NF instances. In other words, nodes that are on the shortest path of many flows are given higher priority to host NF instances, as noted in (**O1**). If the weight of the nodes is equal, then nodes that have higher processing power are given priority over nodes that have lower processing power. Next the NF placement decisions are made for each group of flows.

*B. **Placement Phase***

In this phase, NF instances are placed on the shortest path nodes and/or their neighboring nodes. The type and number of NF instances required to serve each of the groups of flows have been calculated in the initialization phase. The set of *NF types to be placed for a group of flows are ordered according to their popularity, which is measured by the number of flows that require the NF type*. The most popular NFs are prioritized to be placed first, considering (**O3**), with ties broken by prioritizing the NF requiring more processing power. The number of each type of NF to be instantiated is recorded. The placement phase places NF instances for each group of flows.

Algorithm 2 ClusPR's Placement Heuristic

```
1: $Q_n \leftarrow$ priority queue of candidate nodes
2: $Q_{nf} \leftarrow$ priority queue of NF types to be placed
3: $i_v \leftarrow$ number of instances of NF type $v$ to be placed
4: $ActiveNode \leftarrow$ null
5: $n.(list)$ list of node $n \in Q_n$
6: $C \in \{0,1\} \leftarrow C = 1$ if consolidation is used
7: $T \leftarrow$ per NF utilization threshold for consolidation
8: while $Q_{nf}$ not empty do
9:     $v \leftarrow$ NF type from top of $Q_{nf}$
10:    while $Q_n$ not empty do
11:        if $ActiveNode$ and $v \in ActiveNode.(list))$ then
12:            if $C$ & $ActiveNode$ has $v$& $\rho^v_n < T$ then
13:                $i_v = i_v - 1$, Continue to next $v$ in $Q_{nf}$
14:            else if n has resources then
15:                Place $v$ on $ActiveNode$
16:                $i_v = i_v - 1$, Continue to next $v$ in $Q_{nf}$
17:        else
18:            $n \leftarrow$ top of $Q_n$
19:            if $v \in n.(list)$ then
20:                if $C$ &$n$ has $v$ hosted and $\rho^v_n < T$ then
21:                    $i_v = i_v - 1$, continue to next $v$ in $Q_{nf}$
22:                else if n has resources then
23:                    Place $v$ on $n$, $ActiveNode \leftarrow n$
24:                    $i_v = i_v - 1$, Continue to next $v$ in $Q_{nf}$
```

Bin-Packing: The placement heuristic, summarized in Algorithm 2, does a *bin-packing of the ordered NF types on the set of best candidate shortest path nodes and their one hop and two hops neighboring nodes*. An NF instance is placed on a shortest path node if and only if the node has the NF type in its *list*, which contains a list of the NF types needed by the flows whose shortest paths pass through the node. An NF type that is on top of the priority queue of NF types is taken and the queue of "best" candidate nodes is iterated through until a node that has the NF type in its *list* is found. Once a node is found, it is checked if the node has enough processing and memory capacity to support the NF type.

If all the "best" candidate shortest path nodes do not have enough resources to host the NF type, the algorithm checks for one hop neighboring nodes of the shortest path nodes followed by two hops neighboring nodes. Once a node is found the NF is placed and the number of instances of the NF type to be placed is decreased by one. The node will then be regarded as an *active node* for placing the next NF type. Nodes that are more than two hops away from the shortest path nodes could also be considered for hosting NFs but the farther the candidate nodes are from the shortest path nodes, the higher the probability that flows will experience larger path stretch by using NFs hosted on these nodes.

Diversity: The placement *heuristic diversifies the types of NFs placed on a node*. That is, the algorithm prioritizes placing different types of NF instances on one node rather than placing multiple instances of the same type of NF on the node. If different types of NFs are placed on one node, the probability that a flow can get all of the services it requires from one node will be high, as noted in (**O2**). Serving a flow's chain in one node has advantages such as decreasing the communication cost and the delay experienced by the flow.

Next, the following NF type is picked from the queue of NFs and placed on the *active node* provided that the NF is found in its *list*. If not the algorithm returns to the queue of the nodes, and looks for another node following the same steps as above. After placing one instance of all types of NFs, the algorithm returns back to the top of the queue of NFs and places the second instances. This process is repeated until all the instances of all NF types are placed.

Consolidation: The aim of the consolidation step is to share placed NF instances among groups of flows to facilitate better utilization of the NF instances. If consolidation is applied, before placing an NF instance at a given node, the algorithm checks if the node has already hosted this type of NF for the other groups of flows. If the estimated utilization of the instance already placed on the node, ρ_n^v, is below a given threshold (e.g.,50%), it is assumed that the instance has enough available capacity to host the flows in the other group as well so a new instance will not be instantiated. Consolidation will result in the instantiation of fewer number of instances.

Higher values of the threshold encourage consolidation thus leading to the instantiation of less number of NF instances. However, this has a risk of increased path stretch and rejection of flows. Comparatively lower threshold values discourage consolidation. If the *first approach* (initialization phase) is used to decide the number of instances, then the instances created are already the minimum number of instances needed to serve the set of flows so *the consolidation step should not be applied.*

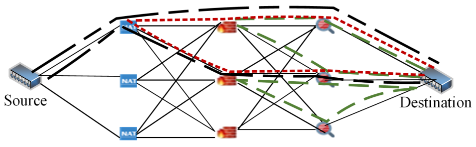


Fig. 2: Example of paths in $\mathcal{R}_{n_j}^k$ and $\mathcal{R}$ for k=2

C. *Routing Phase*

The routing phase is responsible for making flow-level decisions of assigning NF instances to flows and routing flows end-to-end i.e., from their source node through the assigned NF instances of their service chain and finally to their destination node. In making these decisions, two objectives are considered: *satisfying delay requirements of flows* and *load balancing among NF instances*.

A flow's routing problem is *modeled as a multi-stage graph* in which the stages of the multi-stage graph represent the services in the service chain of the flow. The vertexes of a stage of the graph represent the NF instances the flow can be assigned to. At each stage, a flow can be assigned to one of the NF instances that are placed for its group. These are the NF instances placed on the shortest path nodes and their neighboring nodes of the flow's group.

For constructing the multi-stage graph, the costs on the links of the graph also need to be calculated. The costs can be calculated using shortest path algorithms such as Dijkstra's algorithm. The shortest path costs of the links from the source node to the nodes hosting the first NF instance of the chain and the links from the destination node to nodes hosting the last NF of the service chain need to be calculated for each of the flows. The costs of the links between the stages (nodes hosting NFs in the chain) are calculated once, which decreases the computational complexity of constructing the graph. We propose a new algorithm that is based on the ideas of dynamic programming and incorporates novel methods to enable solving the flow routing problem considering both end-to-end delay and the utilization level of instances. Before explaining the algorithm, the dynamic programming based shortest path algorithm is explained for completeness.

To formulate the dynamic program, two distance notations are adopted: $C(n, n^{'})$ and $D(n_j, d_f)$. $C(n, n^{'})$ is used to represent the cost of the shortest distance between nodes n and $n^{'}$ that belong to two consecutive stages (NFs in the chain). e.g., $C(s_f, n_1)$ represents the cost of the shortest distance between the source node of flow f (s_f) and node n_1 that hosts the 1^{st} service instance. $D(n_j, d_f)$ represents the shortest distance between node n_j that is hosting the j^{th} service of the flow to the destination node, e.g., $D(n_2, d_f)$ represents the shortest distance from node n_2 that is hosting the 2^{nd} service to the destination node (d_f).

The dynamic program formulation is given as

$$D(s_f, d_f) = \min_{n_1 \in \mathcal{N}_1} (C(s_f, n_1) + D(n_1, d_f)) \tag{22}$$

$$D(n_j, d_f) = \min_{n_{j+1} \in \mathcal{N}_{j+1}} (C(n_j, n_{(j+1)}) + D(n_{(j+1)}, d_f)) \tag{23}$$

$\mathcal{N}_j$ is the set of nodes that are on the shortest path and one hop and two hops away from the shortest path and are hosting the flow's j^{th} service type for the group the flow belongs to. The dynamic program is solved starting from the destination node until the source node is reached.

*1) **The proposed flow routing algorithm**:* The objectives of the proposed flow routing algorithm are to satisfy the delay requirement of the flow and balance the load among NF instances. To achieve these, the algorithm first *(1) finds a set of routes that satisfy the delay requirement of the flow*, then (2) out of these paths a flow is assigned to a path that has the *minimum maximum NF utilization (to balance the load among NF instances).*

To find a set of routes that could potentially satisfy the delay requirement of the flow, k shortest paths are saved from a node in a stage of a graph to the destination. That is the distance $D(n_j, d_f)$, which represents the shortest distance between a node n_j that is hosting the j^{th} service of the flow to the destination node, is extended to a set of paths, $\mathcal{R}^k_{n_j}$, which has k elements (k shortest paths from a node n_j of the j^{th} stage to the destination). Fig. 2 shows an example of the paths, which are highlighted, saved for k=2. $\mathcal{R}_{n_j}$ represents a set of paths from a node n_j that is hosting the j^{th} service of flow f to the destination node, so $\mathcal{R}^k_{n_j} \subseteq \mathcal{R}_{n_j}$.

Nodes found in the last stage of the multi-stage graph have a direct link with the destination node, so for each node in the last stage, $n_J \in \mathcal{N}_J$, $\mathcal{R}^k_{n_J}$ will have one element only. The distance from nodes in the last stage, J, to the destination node i.e., $D(n_J, d_f)$) is calculated using shortest path algorithms. A set of paths from nodes in the $J-1$ stage to the destination node ($\mathcal{R}_{n_{J-1}}$) can be calculated as:

$$\mathcal{R}_{n_{J-1}} = \{C(n_{J-1}, n_J) + D(n_J, d_f) : \\ n_J \in \mathcal{N}_J, n_{J-1} \in \mathcal{N}_{J-1}\} \tag{24}$$

The set of k shortest paths from a node in the $J-1$ stage to the destination node ($\mathcal{R}^k_{n_{J-1}}$) is taken from the set $\mathcal{R}_{n_{J-1}}$. For a node in a stage $j \in \{1 \dots J-2\}$, a set of distances ($\mathcal{R}_{n_j}$) can be calculated as

$$\mathcal{R}_{n_j} = \{C(n_j, n_{j+1}) + D(n_{j+1}, d_f) : \\ D(n_{j+1}, d_f) \in \mathcal{R}^k_{n_{j+1}}, n_{j+1} \in \mathcal{N}_{j+1}, n_j \in \mathcal{N}_j\} \tag{25}$$

The set $\mathcal{R}_{n_j}$ is constructed by using the set of k shortest paths to the destination saved in the $j+1^{th}$ stage (i.e., $\mathcal{R}^k_{n_{j+1}}$) and costs between nodes in the stages j and $j+1$ of the multi-stage graph, $C(n_j, n_{j+1})$. Similarly, a set of k shortest distances ($\mathcal{R}^k_{n_j}$) is found from the set $\mathcal{R}_{n_j}$ for each node $n_j \in \mathcal{N}_j$. That is the k shortest paths from all NFs of a stage to the destination node are calculated for the stages one to $J-1$. Finally a set of source to destination end-to-end paths are calculated as:

$$\mathcal{R} = \{C(s_f, n_1) + D(n_1, d_f) : \\ D(n_1, d_f) \in \mathcal{R}^k_{n_1}, n_1 \in \mathcal{N}_1\} \tag{26}$$

If there are N_1 number of nodes in $\mathcal{N}_1$ i.e., the first stage of the graph, with each node having k shortest paths to the destination, there will be in total kN_1 number of source to destination end-to-end paths in the set $\mathcal{R}$.

Out of these paths in $\mathcal{R}$, the set of paths that are able to fulfill the delay requirement of the flow ($\mathcal{R}^d$) are identified. Then, the objective of balancing the load among the NF instances is considered by adopting the min-max fairness. The maximum utilization of the NF instances on each of the routes in $\mathcal{R}^d$ is calculated. The route that has the minimum maximum NF utilization is then chosen for serving and routing the flow. In the situation where there are no routes that can satisfy the delay requirement of the flows ($\mathcal{R}^d$ is empty), the flow is assigned to a route that has the minimum end-to-end delay.

Theorem 3. *ClusPR has a complexity of $O(FN_gL\log N)$), where N_g is the average number of shortest path and neighboring nodes per group. F, N and L are the number of flows, nodes and links respectively.*

Proof: refer to Appendix C.

V. Online Placement and Flow Routing: incremental ClusPR(iClusPR)

ClusPR is an offline algorithm that performs NF placement and flow routing decisions for a set ($\mathcal{F}$) of flows, i.e., the information of all flows is known to ClusPR beforehand. In an online environment, flows will arrive sequentially so resource allocation decisions need to be made for the flows that arrive without knowledge of future incoming flows. iClusPR is an online NFV resource allocation algorithm. It is developed based on ClusPR so it has a design similar to ClusPR (shown in Fig. 1). iClusPR performs dynamic scaling specifically horizontal scaling that is adjusting the number of NF instances depending on the traffic demand. iClusPR makes resource allocation decisions on a time slot basis. That is flows that arrive at a given time will be assigned resources at the subsequent decision time slot. The modifications made in iClusPR for each of the modules are explained below:

A. Initialization phase

1) Clustering: The clustering module of iClusPR serves the same purpose as in ClusPR, that is to group flows based on the proximity of their path. The same clustering algorithm as in ClusPR is used. iClusPR clusters access nodes once when the algorithm is run for the first time on a network topology. Upon arrival of flows, the clustering module simply groups flows based on their source and destination nodes. Then, the NF instances needed to serve each group of flows are calculated using the second approach, which is explained in the initialization phase of ClusPR.

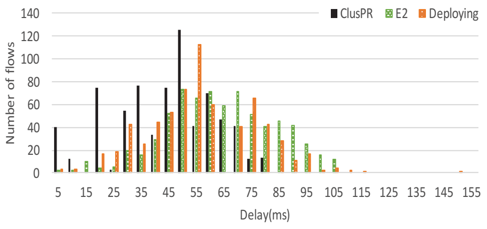
2) Shortest Path: This module of iClusPR is similar to its counterpart in ClusPR and it extracts path information by identifying nodes that are on the shortest path of flows. This calculation needs to be performed upon arrival of flows.

B. Placement phase

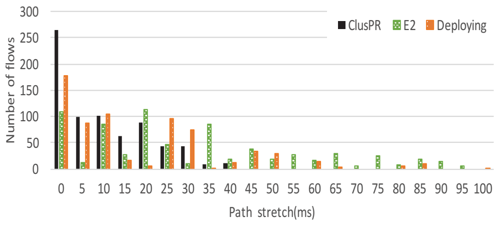
This phase performs *dynamic scaling* by adjusting the number of NF instances instantiated in the network. It takes as input the shortest path nodes and their one hop and two hops neighboring nodes, the NF instances created on these nodes as well as the type and number of NF instances needed to serve each group of flows.

The NF instances instantiated in the previous decision slots can serve the incoming flows as well provided that they have enough available capacity. iClusPR uses a threshold based approach to decide if an NF instance is able to serve the incoming flows or not. That is NF instances whose residual or available capacity is above the threshold value are assumed to have enough available capacity for hosting the incoming flows. Higher threshold values encourage the instantiation of new NF instances and might lead to overprovisioning or underutilization of resources. On the other hand, lower threshold values encourage the use of existing NF instances but could result in over-utilization of resources and rejection of flows.

Similar to ClusPR's placement heuristics, iClusPR orders the set of NF types needed to be created based on their popularity, which is measured by the number of flows that need the NF type. The most popular NF type is prioritized to

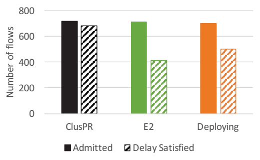


(a) Distribution of total delay

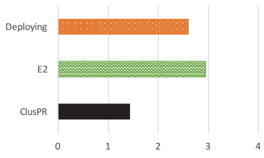


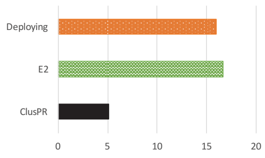
(b) Distribution of path stretch

Fig. 3: Delay performance with service chain length=2



(a) Network utilization, chain = 2


(b) Average delay, chain = 2


(c) Worst delay, chain = 2

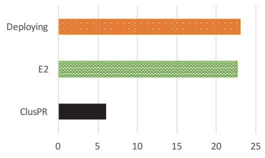

(d) Worst delay, chain= 4

Fig. 4: Network utilization, average and worst-case normalized total delays for different chain lengths

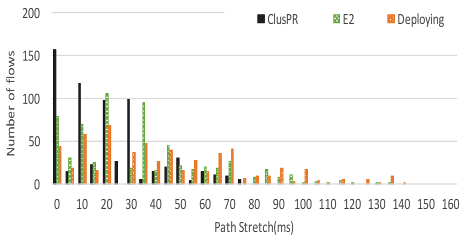


Fig. 5: Path stretch distribution, chain length = 4

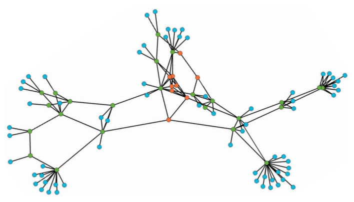


Fig. 6: Rocketfuel topology AS 1221:100 nodes and 294 links

be placed first. The NF type that is on the top of the NF queue is picked first and the *placement heuristic checks whether there is an existing NF of the same type whose residual capacity is above the threshold specified*. If so, the algorithm goes to the next NF type without instantiating a new instance. If there are no existing NF instances of the same type, a node that is on top of the selected nodes queue is picked and a new instance of the NF type is instantiated following the same steps as in ClusPR's placement heuristic. Besides creating new NF instances, an instance might also be removed from the network if all the flows it was serving have departed.

C. Routing phase

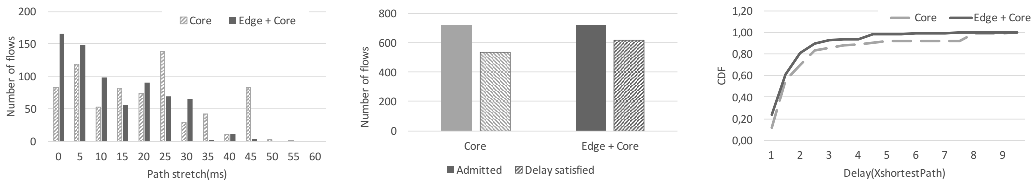
In this phase, flows are assigned NF instances and routed end-to-end that is from their source node through the NF instances of its chain finally to their destination node. The same algorithm as in the routing phase of ClusPR is used.

VI. Experimental Results

We analyze the performance of ClusPR and iClusPR extensively. ClusPR's performance is also compared with two alternatives, E2 [8] and [11] (referred to as "Deploying" here), on realistic networks. We report results on experiments with a practical ISP network topology, the Rocketfuel [23] topology AS 1221 shown in Fig. 6 used as a test network. The nodes in the topology are classified as "access" (in blue), "edge" (green) and "core" (orange) nodes, in a manner similar to [24]. NFs are considered to be hosted on (or adjacent to) edge and core nodes. It is assumed that each host has 4 CPU cores and 8GB memory. Every NF instance requires one CPU core and 2GB memory, with a service rate of 10Mbps. There are five types of NFs (e.g., Firewall, DPI, NAT, IDS, and Proxy).

All the links have a capacity of 1 Gbps, and the delays on the links are: access-edge: 3 ms; edge-core: 10 ms; core-core: 40 ms. The source and destination nodes of flows as well as the services required by the flows are generated randomly. The arrival rate of each flow is assumed to follow a log-normal distribution [25] with an average rate of 0.5 Mbps. The length of the service chain for each flow is assumed to vary in the range of 2 to 4 NFs and the service types in the chain for each flow are selected randomly.

The performance metrics used are total delay, path stretch (measured as the difference between the total delay and shortest path delay), network utilization (measured by the number of flows admitted and also by those whose delay requirement is satisfied), number of NF instances created and the per-NF utilization level. These performance metrics are compared for different setups such as variable chain length and distribution of node processing capacity.



(a) Distribution of path stretch (b) Network utilization (c) CDF of the Normalized Delay

Fig. 7: Effect of using edge computing nodes

A. *Evaluation of ClusPR*

Total Delay and Path Stretch:

The total delay of a flow is measured as the summation of the delays on the links it is routed through. Fig. 3 shows the total delay and path stretch distributions with 720 flows that have a two NFs long service chain. Both E2 and ClusPR instantiated 74 instances, which is the minimum number of instances needed for the flows.

ClusPR has a shorter path stretch compared to both Deploying [11] and E2 for the *same number of instances*. The performance gain is partly because ClusPR uses the flows' path information in NF placement decision-making and it diversifies the type of NFs placed on a node, thus increasing the probability that a flow can get all of the services of its chain at one node.

Average and Worst-case Delays: Fig. 4b and 4c show the average and worst-case total delays, respectively. They are normalized with respect to the shortest path delay of flows. ClusPR is able to achieve a worst-case normalized delay that is $10\times$ less than the worst-case normalized delay of E2 and Deploying. The average normalized delay of ClusPR is $1.2-1.6\times$ less compared to E2 and Deploying.

Network Utilization: To analyze the delay satisfaction of flows, flows are set to have a specified delay requirement in terms of the maximum normalized total delay that they can tolerate. The normalized delay requirement of flows is assumed to be uniformly distributed between $1-2.5\times$ their shortest path delay.

Fig. 4a shows the number of flows that are admitted out of the 720 flows and those whose delay requirement is satisfied. ClusPR, E2 and Deploying achieve similar network utilization in terms of the number of admitted flows, but their delay performance differs considerably, which also results in a noticeable difference in terms of the number of flows whose delay is satisfied. The delay difference is due to the following underlying reason. For E2, delay or path stretch is not considered in the heuristic. Deploying, on the other hand, prioritizes maximizing network utilization and does not balance the load across NF instances. ClusPR satisfies the delay requirement of 95% of the flows while E2 and Deploying managed to fulfill the delay requirement of 60% and 70% of the flows, respectively. ClusPR satisfied the delay requirement of $25-35\%$ more flows compared to E2 and Deploying.

Effect of Service Chain Length: We now analyze the effect of the service chain length. Fig. 5 demonstrates the path stretch distribution of 650 flows with a service chain length of four. Both E2 and ClusPR instantiated 132 NF instances for serving the flows.

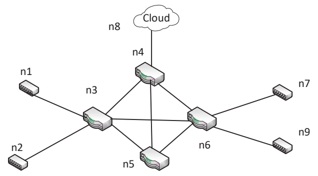
In comparison to the path stretch experienced by flows with service chain length=2 (Fig. 3b), flows with a service chain length of four experienced a larger path stretch. This is because flows with longer service chains need to traverse through multiple NF instances which might not be co-located in the same node. The worst-case delay is shown in Fig. 4d. Compared to the worst-case delay experienced for service chain length of 2 (Fig. 4c), ClusPR's worst-case delay for a chain length of 4 has increased slightly from $5\times$ to $6\times$ the shortest path delay. E2 and Deploying have observed $6\times$ and $7\times$ increase in the worst-case normalized delay. Thus, ClusPR can adapt better to the change in the service chain length compared to both E2 and Deploying.

Effect of edge computing nodes: In this section, the effect of using edge computing nodes as hosts of NF instances is analyzed. Two network setups are considered. In the first setup the processing power is concentrated in three centralized (core) clouds. Each of the clouds has 44 cores. Thus, in total there are 132 CPU cores in the network. In the second setup, some of the processing power of the central clouds is distributed to the edge nodes. The three central cloud nodes have 24 cores each while 30 edge nodes have two cores each. We will refer to this setup as "Edge + Core" and the first setup as "Core". Thus, in total the "Edge + Core" network setup will also have 132 CPU cores.

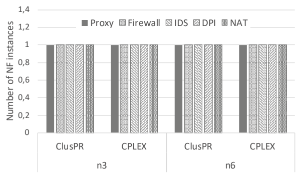
Fig. 7a shows the path stretch observed by 700 flows in the "Edge + core" and "Core" network setups. The "Edge + core" setup has a better performance as more flows experience zero or a small amount of path stretch compared to the "Core" setup. As can be inferred from the CDF of the normalized delay shown in Fig. 7c, the "Edge + core" setup has a smaller worst-case and average total delays. The number of flows that are admitted and those whose delay requirement is satisfied is shown in Fig. 7b. The delay requirement of 75% and 86% of the flows is satisfied in the "Core" and "Edge + core" setups respectively. These results demonstrate that hosting NF instances on edge computing nodes also has a significant performance gain by decreasing the latency.

B. *Effect of load balancing across NF instances*

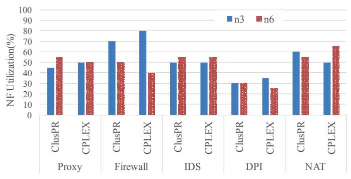
ClusPR's flow routing algorithm balances the load among the instantiated NF instances. In this section, ClusPR's load balancing approach is analyzed and compared with the no load balancing approach, which assigns to flows NFs that are on



(a) Test network

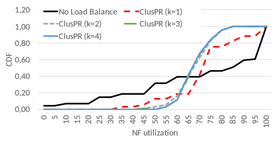


(b) NFs on nodes n3 & n6

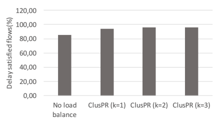


(c) Per-NF utilization

Fig. 8: Comparison between ClusPR and CPLEX



(a) CDF of NFs utilization



(b) Network utilization

Fig. 9: Effect of load balancing

the shortest path of the multi-stage graph model of the flow, provided the NFs are not 100% utilized. ClusPR's flow routing algorithm saves k shortest paths from a node at a given stage to the destination node, then it balances the load considering the multiple paths saved. k is an algorithmic parameter and the impact of the value of k is also assessed.

In Fig. 9a the utilization of instances when no load balancing is applied is compared with ClusPR with k varying from one to four. When no load balancing is applied, some of the instances have very low utilization while some other instances are highly utilized, going up to 100%. For ClusPR with k=1, there is a better distribution of load among the instances. For higher values of k, the load is balanced across the instances with most instances having a utilization between 65%-75%. There is only a slight difference in the utilization level of instances for k more than 3. As a result of the load balancing, ClusPR is able to satisfy the delay requirement of 8% more flows compared to the no-load balancing approach (Fig. 9b).

C. *Comparison of ClusPR and ILP Model*

The performance of ClusPR is compared with the proposed ILP model (solved using CPLEX) for a small Test network topology with 9 nodes and 11 links shown in Fig. 8a. Nodes n3, n4, n5, n6 and n8 are able to host NF instances and have a processing capacity of 5,4,4,5 and 10 cores, respectively.

The number and type of instances instantiated for serving 50 flows are shown in Fig. 8b. Two nodes, node n3 and n6, which are on the shortest path of flows are chosen to host NFs by both CPLEX and ClusPR. As can be seen from the figure, ClusPR has created the same number and type of instances as the optimal CPLEX solution. The utilization of the instances is shown in Fig. 8c. ClusPR balances the load across the NF instances and the utilization is only very slightly different from the utilization of the instances in the CPLEX solution. Both ClusPR and CPLEX are able to satisfy the delay requirement of all 50 flows. For the execution time, CPLEX takes 1 hour while ClusPR takes less than 1 second. to get the solution, on the same computer. This indicates that, ClusPR is able to reach a near optimal result, but is much faster.

D. *Evaluation of iClusPR*

The performance of iClusPR is analyzed and compared with ClusPR, and the effect of the parameter, α, which is a per-NF utilization threshold value, is assessed. In this evaluation, a flow-level simulation was performed.

Simulation setup: The flow arrival process is assumed to be Poisson with an average arrival rate of 1 flow per time unit. The sojourn time of flows is exponentially distributed with an average of 700 time units. The decision time slot is assumed to be 40 time units long. Under these settings, the network can be modeled as an $M/M/\infty$ system. From queuing theory, theoretically, the average number of flows expected in the network is 700.

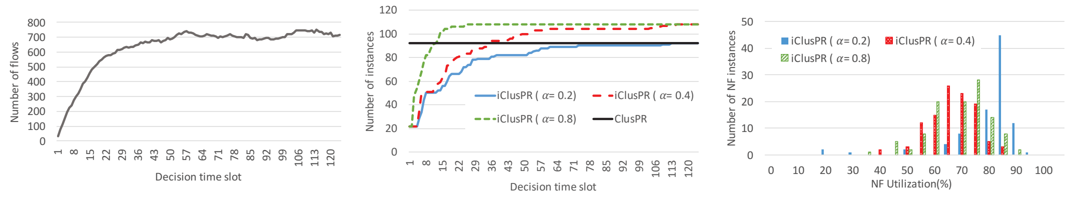
Fig. 10a shows the number of flows in the system at different decision time slots. In total, by time slot 125, 5000 flows have arrived. As expected, the number of flows in the system converges to the theoretical average value i.e., 700. The number of NF instances created for different values of the threshold value (α) is shown in Fig. 10b. For $\alpha = 0.8$, a new NF instance will be instantiated if the existing NF instances have less than 80% available capacity. Thus, the larger the value of α the more the number of instances created. iClusPR also balances the load across the NF instances as can be seen from Fig. 10c. Figs. 11a and 11b show the percentage of flows whose delay requirement is satisfied and the worst-case delays, respectively, for different threshold values. For $\alpha = 0.2$, the network has the lowest performance of all and $\alpha = 0.4$ and 0.8 have almost similar performance. An implication is, α values that are in the middle e.g., $\alpha = 0.4$ or 0.5, give a good trade-off between the number of NF instances and the network performance (delay, utilization).

VII. Related Work and Discussion

In the literature, a number of approaches have been proposed for the NFV-RA problem. A detailed survey can be found in [26]. In the following, we review the most related and recent works.

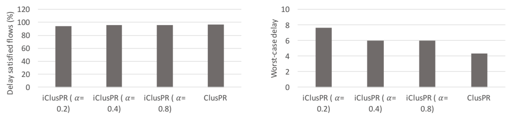
ILP models for the joint NF placement and flow routing problem have been presented in papers such as [27], [28] and [29]. In addition to the models, heuristic algorithms (based on the models) have been proposed in [27] and [28] and a greedy heuristic is proposed in [29]. A shortcoming of the [27] and [29] models is that they do not keep the routing order among services of a chain. In addition, the ILP models are generally not scalable due to the complexity of the problem.

Recently, heurisitc approaches have been proposed to tackle the scalability problem associated with the ILP models. Here



(a) Number of flows (b) Number of instances (c) Per-NF utilization

Fig. 10: Evalutaion of iClusPR for different values of per-NF utilization theshold(α)



(a) Network utilization (b) Worst-case Delay

Fig. 11: Evalutaion of iClusPR

we broadly divide the algorithms into two classes. The first class includes algorithms that avoid path stretch by serving the flows in their path. For example, the resource allocation algorithm in, CoMb [9], [10] and centrality-based heuristics such as [30] and [31] belong in this class. In CoMb, a flow is constrained to use NF instances running in the same node that is found on its path. However, the CoMb approach can considerably limit the utilization of the network since flows are constrained to stay on their path and use a single node for all their services. Relatively, the centrality-based heuristic in [31] has a relaxed restriction as it allows a flow to use NFs placed on more than one node, but still the nodes have to be located on the shortest path of the flow.

In the second class are algorithms that try to increase the utilization of the network disregarding the path stretch. For examples, algorithms proposed in [8], [11], [12], [32] and [33] belong in this class. In [11], referred to as Deploying in this paper, an algorithm that tries to make better use of network resources by promoting flows to reuse instances which have been created instead of instantiating new ones is proposed. Another heuristic approach is the E2 framework [8] which is developed for allocating NF instances and routing flows inside a central office or small data centers. The placement is modeled as a graph partitioning problem and solved using a modified Kernighan-Lin heuristic. Flows are assigned to NFs balancing load across the NF instances.

The proposed schemes, ClusPR and iClusPR, take an approach that can be regarded as being in the middle of these two classes. They do not impose strict restrictions on flows to not deviate from their shortest path. This is because flows might have a relaxed delay requirement which may not be violated even if they deviate from their shortest path. They also do not disregard the effect of path stretch as methods in the second class. ClusPR and iClusPR rather find a balance between minimizing the path stretch, maximizing network utilization and balancing the load among the NF instances. ClusPR and iClusPR are targeted at general networks where resources are distributed in the networks. For networks that have minimal delay between the nodes, as for example in a data-center network, we expect that the performance gain with ClusPR will not be as significant, compared to [8] and [21]. However, ClusPR and iClusPR address the more general problem of having NFs placed at diverse locations, including multiple data-centers across a WAN, which would be required to address scale and diversity typically observed in service provider networks. We believe that ClusPR's ability to consider the trade-off across multiple dimensions will prove invaluable in production networks

VIII. Conclusion

The flexibility brought about by NFV can potentially change the way networks are managed and services are provisioned. However, efficient resource allocation algorithms are needed to instantiate NF instances when and where needed, and route flows through them accordingly. A comprehensive ILP model is provided for the NFV-RA problem. Based on the useful insights obtained from the optimal solution of the ILP model, we develop an offline heuristic algorithm, ClusPR, that is scalable and balances across multiple objectives. In addition, an online algorithm, iClusPR, that dynamically adjusts the number of NFs depending on the traffic demand and network state is presented. By factoring in information about the path of flows into the NF placement decision making and diversifying the type of NFs placed on a node, ClusPR and iClusPR are able to considerably minimize the path stretch and maximize the network utilization while balancing the load across NF instances. Compared to the state-of-the-art approaches, ClusPR manages to decrease the average normalized delay by a factor of $1.2-1.6\times$ and the worst-case delay by $10\times$, while admitting the same or slightly larger number of flows. At the same time, ClusPR satisfies the delay requirement of 25-35% more flows and balances the load across NF instances. The online algorithm iClusPR is also able to perform dynamic NF scaling while having performance that is comparable to that of ClusPR.

Appendix A

Proof of Theorem 1:

In order to have a stable system, a server (NF instance) should not be loaded more than its service rate. The aggregate arrival rate of flows that require an NF type v is equal to $F^v\lambda$. The total service rate of v type NF instances should be more than the aggregate arrival rate of the flows, that is $F^v\lambda < \mathcal{I}^v_{min}\mu^v$. Where $\mathcal{I}^v_{min}$ is the minimum number of v type NF instances. Thus, $\mathcal{I}^v_{min} = \lceil \frac{F^v\lambda}{\mu^v} \rceil$

■

APPENDIX B

Proof of Theorem 2:

For each of the group of flows $g \in \mathcal{G}$, ClusPR calculates the minimum number of NF instances needed to serve the flows using Theorem 1. The maximum total number of NF instances instantiated is a summation of the minimum number of NF instances calculated for each of the groups, that is $\mathcal{I}^v_{max} = \sum_{\forall g} \lceil \frac{F^v_g \lambda}{\mu^v} \rceil$, F^v_g is the number of flows in group g that require NF type v. The maximum value $\mathcal{I}^v_{max}$ can take is $\sum_{\forall g}(\frac{F^v_g \lambda}{\mu^v} + 1)$. Where $\frac{F^v_g \lambda}{\mu^v}$ is an integer quotient of the float division. Since there are a maximum of $\frac{k^2+k}{2}$ number of groups.

$$\mathcal{I}^v_{max} = \sum_{\forall g}(\frac{F^v_g \lambda}{\mu^v} + 1) = \frac{\sum_{\forall g} F^v_g \lambda}{\mu^v} + \frac{k^2+k}{2}. \quad (27)$$

since $\sum_{\forall g} F^v_g = F^v$,

$$\mathcal{I}^v_{max} = \frac{F^v \lambda}{\mu^v} + \frac{k^2+k}{2} = \mathcal{I}^v_{min} + \frac{k^2+k}{2} \quad (28)$$

According to Theorem 1, the minimum number of instances that need to be instantiated to serve the set of flows ($\mathcal{F}$) is $\mathcal{I}^v_{min}$. Thus, the number of NF instances instantiated by ClusPR($\mathcal{I}^v$) is bounded by

$$\mathcal{I}^v_{min} \leq \mathcal{I}^v \leq \mathcal{I}^v_{min} + \frac{k^2+k}{2} \quad (29)$$

■

APPENDIX C

Proof of Theorem 3:

The initialization phase has two modules these are clustering and shortest path. The clustering module uses Kruskal's algorithm which has a complexity of O(Llog$L$). The shortest path utilizes Dijkstra's shortest path algorithm which has a complexity of O($L \log N$). Thus the complexity of the initialization phase is *O($L \log L$ + $L \log N$)*.

The complexity of the placement heuristics depends on, the number of flow groups, number of instances to be placed and the number of nodes that can host NFs. From Theorem 1, the number of instances to be placed can roughly be approximated by the number of flows (F). The placement heuristic is run for each of the group of flows. For each group, ClusPR utilizes the nodes that are on the shortest path of flows and their neighboring nodes as candidate nodes for placing NF instances. Let N_g be the number of these candidate nodes and G be the total number of groups. Thus, the complexity of the placement heuristics will be *O(GFN_g)*.

The routing of a flow is modeled as a multistage graph. The maximum number of vertices of the graph is equal to JN_g where J the number of stages of the graph and N_g is the number of candidate nodes. The number of edges between the stages of the graph is equal to $N_g(N_g - 1)(J - 1)$. In addition, there will be $2N_g$ edges between the source node of the flow and the nodes in the first stage and the destination node and nodes in the last stage of the graph. Simplifying, in total the multi-stage graph will have $N_g^2 + N_g$ edges. Dijkstra's shortest path algorithm is used to find the cost of the edges, which will have a complexity O($L \log N(FN_g + N_g^2)$).

ClusPR's routing algorithm has complexity proportional to the complexity of a dynamic programming shortest path algorithm whose complexity is in the order of the summation of the number of edges and vertices of the multi-stage graph. Since the graph is constructed for each of the flows, the complexity of solving the multi-stage graph for all of the flows will be O($F(N_g^2 + N_g + JN_g)$). Thus, the complexity of the routing algorithms is *O($L \log N(FN_g + N_g^2) + F(N_g^2 + N_g + JN_g)$)*. The complexity of ClusPR will be a summation of the complexity of the three phases and can be simplified to $O(FN_g L \log N)$). ■

ACKNOWLEDGMENT

The work for this paper was performed in the context of the EU FP7 Marie Curie Actions project Grant Agreement No. 607584 (the CleanSky project).

REFERENCES

[1] Y. Woldeyohannes *et al.*, "A scalable resource allocation scheme for NFV: Balancing utilization and path stretch," in *Innovations in Clouds, Internet and Networks (ICIN), 2018 21th Conference on*. IEEE, 2018.

[2] J. Sherry *et al.*, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[3] W. Zhang *et al.*, "Opennetvm: A platform for high performance network service chains," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*. ACM, 2016, pp. 26–31.

[4] Y. Zhang *et al.*, "Steering: A software-defined networking for inline service chaining," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. IEEE, 2013, pp. 1–10.

[5] A. L. Andreas Lemke, "Why service providers need an NFV platform: Strategic white paper," Tech. Rep., January, 2015.

[6] A. Gember *et al.*, "Stratos: A network-aware orchestration layer for middleboxes in the cloud," Technical Report, Tech. Rep., 2013.

[7] M. Xia *et al.*, "Network function placement for NFV chaining in packet/optical data centers," in *Optical Communication (ECOC), 2014 European Conference on*. IEEE, 2014, pp. 1–3.

[8] S. Palkar *et al.*, "E2: a framework for NFV applications," in *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 2015.

[9] V. Sekar *et al.*, "Design and implementation of a consolidated middlebox architecture," in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012.

[10] Y. Sang *et al.*, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proceedings of IEEE INFOCOM 2017*. IEEE, 2017, pp. 829–837.

[11] T.-W. Kuo *et al.*, "Deploying chains of virtual network functions: On the relation between link and server usage," in *INFOCOM, 2016 Proceedings IEEE*. IEEE, 2016.

[12] S. Khebbache *et al.*, "Virtualized network functions chaining and routing algorithms," *Computer Networks*, vol. 114, pp. 95–110, 2017.

[13] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 7–13.

[14] Z. A. Qazi *et al.*, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 27–38.

[15] W. Zhang *et al.*, "SDNFV: flexible and dynamic software defined control of an application-and flow-aware data plane," in *Proceedings of the 17th International Middleware Conference*. ACM, 2016, p. 2.

[16] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[17] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[18] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 68–86, 1971.

[19] O. Grygorash *et al.*, "Minimum spanning tree based clustering algorithms," in *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*. IEEE, 2006, pp. 73–81.

[20] N. Bolshakova and F. Azuaje, “Cluster validation techniques for genome expression data,” *Signal processing*, vol. 83, no. 4, pp. 825–833, 2003.

[21] J. C. Dunn, “A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters,” *Journal of cybernetics*, vol. 3, no. 3, pp. 32–57, 1974.

[22] U. Maulik *et al.*, “Performance evaluation of some clustering algorithms and validity indices,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.

[23] N. Spring, R. Mahajan, and D. Wetherall, “Measuring isp topologies with rocketfuel,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[24] A. Afanasyev *et al.*, “Interest flooding attack and countermeasures in named data networking,” in *IFIP Networking Conference, 2013*. IEEE, 2013, pp. 1–9.

[25] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, “On the characteristics and origins of internet flow rates,” in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 309–322.

[26] J. G. Herrera and J. F. Botero, “Resource allocation in NFV: A comprehensive survey,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[27] B. Addis *et al.*, “Virtual network functions placement and routing optimization,” in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 171–177.

[28] A. Mohammadkhan *et al.*, “Virtual function placement and traffic steering in flexible and dynamic software defined networks,” in *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.

[29] M. C. Luizelli *et al.*, “Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions,” in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 98–106.

[30] M. Bouet *et al.*, “Cost-based placement of vdpi functions in NFV infrastructures,” *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[31] S. Ahvar *et al.*, “CCVP: Cost-efficient centrality-based VNF placement and chaining algorithm for network service provisioning,” in *Network Softwarization (NetSoft), 2017 IEEE Conference on*.

[32] M. C. Luizelli *et al.*, “A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining,” *Computer Communications*, 2016.

[33] M. Mechtri *et al.*, “A scalable algorithm for the placement of service function chains,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533–546, 2016.

Paper C

Measures for Network Structural Dependency Analysis

Yordanos T. Woldeyohannes and Yuming Jiang

IEEE Communications Letters 22.10 (2018): 2052-2055.

In the published version of paperC, in equation (10) N-1 is used in the denominator rather than N-2. However, since there are N -2 elements in $\mathcal{N} \quad ^n/i \neq j$, for getting the value 1 in total dependency, the denominator should be N-2. Note: the results in table II and III are obtained by using N - 2.

As a result of the change in equation (10), the N-1 in equations (13) and (14) denominator is replaced by N-2, and the $\frac{N}{N \quad 1}$ below equation (14) is replaced by $\frac{N^2}{(N \quad 1)(N \quad 2)}$

Measures for Network Structural Dependency Analysis

Yordanos T. Woldeyohannes, Yuming Jiang

***Abstract*—A set of new measures for network structural dependency analysis is introduced. These measures are based on geodesic distance, which is the number of links in a shortest path. They capture the structural dependency effect at the path level, the node level and the overall network level, and hence can be used to index such dependencies. Unlike the related literature measures, a novel aspect of the proposed measures is that the impact of network fragmentation caused by a node failure is taken into explicit consideration in deciding the structural dependency effect. As a result, when applied to critical node identification in a network, the proposed measures give results that are more in line with intuition.**

I. Introduction

Networked systems such as communication networks have become an indispensable part of our daily life. As a consequence, failure of such a system even for short time, e.g. a few minutes or hours, could already be unacceptable let alone for longer time. However, network component failures (e.g. due to hardware, software and communication failures) are often. For a network, its inherent structural dependencies among nodes imply that the impact of one node's failure on the services provided by the network may significantly differ from that of another node's failure. Here arises a fundamental question, referred to as the *structural dependency impact problem* in this work, which is, *what measures may be used to assess the network structural dependency-caused impact*?

The purpose of this paper is to propose an answer to the *structural dependency impact problem*, following the idea that "the importance of a node is related to the ability of the network to respond to the deactivation of the node from the network" [1]. To this aim, a new set of *geodesic distance* based information centrality measures will be introduced, termed as *dependency indexes*. Specifically, these measures are the *path dependency index*, the *node dependency index* and the *network dependency index*. They respectively quantify the impact of a node's failure, at the path level on information communication from one node to another node in the network, at the node level on information communication from one node to other nodes in the network, and at the network level on information communication from any node in the network.

The dependency impact problem is related to the critical node detection problem, which is the problem of finding the most important nodes in a network and has applications in various fields [2]. In communication networks, such applications include network vulnerability analysis [3], critical node discovery [4] and robustness study [5]. In the literature, various measures have been proposed for critical node detection under the concept of *centrality* [6]. The classic centrality measures include node degree, closeness, betweenness and information [5] [7]. However, these measures are generally for the network level, where the impact of structural changes after the node failure at the path level and the node level is not focused.

The most related works are [1] and [8]. In [1], a new category of centrality measures, called *delta centrality*, are introduced. However, as implied by the definition, delta centrality measures only address the dependency impact problem at the network level. In [8], *absolute drop* in reciprocal geodesic distance is used as the basis to quantify the dependency impact, but only at the path level and the node level. In addition, as to be exemplified and analyzed, the dependency measures introduced in [1] and [8] have a strong limitation: While the removal of a node from a network may result in network fragmentation, this effect is not factored in these measures. Addressing this limitation and systematically quantifying the different level dependencies for the dependency impact problem constitute the novelty and contribution of this work.

The rest is organized as follows. First, the unification of the dependency measures in [1] and [8] is proved in Sec. II, where an example showing their limitation will also be given and discussed. Then, the set of new measures are proposed in Sec. III. In Sec. IV, results and *the application of the proposed measures for critical node identification* are demonstrated, compared and discussed. Finally, Sec. V gives the conclusion.

II. Network Model and Existing Measures

A. Network Model

We consider a network $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of the N nodes and $\mathcal{L}$ is the set of the L links. We assume that nodes communicate through their shortest path.

The *information* measure [7] is used as the basis to quantify the influence of a node on a path, another node or the network, or in other words, how a path, another node or the network depends on the node. We use G^{-n} to denote the network that is the same as the original network G but with all links of node n removed, and $\mathcal{N}^{-n}$ to denote the remaining set of $\mathcal{N}$ after excluding node n. By definition, $\mathcal{N}^{-n}$ has $N-1$ nodes.

The concept of *information* between pairs of nodes was originally introduced in [7] as a centrality measure based on the theory of statistical estimation. For shortest path based communication, the *information* measure I_{ij} between node i and node j can be written as the reciprocal of the topological distance d_{ij} between the two nodes, i.e.

$$I_{ij} = \frac{1}{d_{ij}}, \tag{1}$$

where d_{ij} represents the *geodesic distance, i.e. the number of links in a shortest path*, between node i and node j. For a node to itself, it is defined that $d_{ii} = 0$ or $I_{ii} = \infty$; if there is no path between nodes i and j, $d_{ij} = \infty$ or $I_{ij} = 0$.

B. Existing Measures

In [1], to quantify the influence of a node n on the network G, a delta centrality measure, denoted as $\Delta(G|n)$, is introduced as a measure of the relative drop in the network efficiency caused by the deactivation of node n:

$$\Delta(G|n) = \frac{E(G) - E(G^{-n})}{E(G)}, \tag{2}$$

where $E(G)$ denotes the efficiency of the network G which, initially introduced in [9] based on the communication or information efficiency measure, is defined as:

$$E(G) = \frac{1}{N(N-1)} \sum_{\forall i \neq j \in \mathcal{N}} I_{ij}.$$

Note that the delta centrality measure $\Delta(G|n)$ only provides measure at the network level, i.e. how the failure of node n may impact the network.

In [8], a measure of the impact of node n on the path from node i to node j, denoted as $D(i \to j|n)$, is defined as

$$D(i \to j|n) = \frac{1}{d_{ij}} - \frac{1}{d_{ij}^{-n}} = I_{ij} - I_{ij}^{-n}, \tag{3}$$

where d_{ij}^{-n} denotes the geodesic distance between node i and node j in the network G^{-n} and $I_{ij}^{-n} = \frac{1}{d_{ij}^{-n}}$. By the definitions, it is clear that $d_{ij}^{-n} \geq d_{ij}$. Also in [8], based on the path level measure $D(i \to j|n)$, a node level measure, denoted as $D(i|n)$, is introduced which essentially measures the average influence or impact of node n on all the paths from node i to all other nodes, defined as:

$$D(i|n) = \frac{1}{N-1} \sum_{j \in \mathcal{N}^{-n}} D(i \to j|n),$$

which, with (3) applied, can be further written as

$$D(i|n) = \frac{1}{N-1} \sum_{j \in \mathcal{N}^{-n}} (I_{ij} - I_{ij}^{-n}). \tag{4}$$

C. Unification of the Existing Measures

At a first glance, the network level measure $\Delta(G|n)$ seems to be irrelevant to the two dependency measures $D(i \to j|n)$ and $D(i|n)$. In the following, we first extend the latter measures to a network level dependency measure, denoted as $D(G|n)$. Then, we prove the equivalency in ranking nodes between $\Delta(G|n)$ and the new network level dependency measure $D(G|n)$. Through this, the three measures, $\Delta(G|n)$, $D(i \to j|n)$ and $D(i|n)$ are unified.

As defined, $D(i \to j|n)$ is a measure of the influence or impact of node n on the path from node i to node j, based on which $D(i|n)$ essentially measures the average influence of node n on all the paths from node i to all other nodes. Since in (4), i can be any node in $\mathcal{N}$, then by taking average over all N such choices, we can extend the node level measure to a network level measure of the impact to all nodes as:

$$D(G|n) = \frac{1}{N} \sum_{i=1}^{N} D(i|n). \tag{5}$$

The following theorem summarizes the equivalency between $\Delta(G|n)$ and $D(G|n)$.

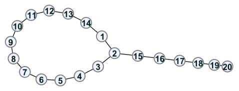


Fig. 1. Tadpole network

TABLE I
PATH DEPENDENCY MEASURES: TADPOLE NETWORK IN FIG. 1

j	3	4	5	8-14	15	16	17	18	19
$D(1 \to j\|2)$	0.42	0.24	0.15	0	0.5	0.66	0.25	0.2	0.17
$DI(1 \to j\|2)$	0.42	0.24	0.15	0	1	1	1	1	1

Theorem 1. *The ranking result of nodes based on $\Delta(G|n)$ is the same as that based on $D(G|n)$.*

Proof. Note that in the definition of $\Delta(G|n)$, $E(G)$ is the same for all nodes. Hence, the ranking result of nodes based on $\Delta(G|n)$ is the same as that based on $E(G) - E(G^{-n})$, which after applying the definition of efficiency becomes

$$E(G) - E(G^{-n}) = \frac{1}{N(N-1)} \sum_{\forall i \neq j} (I_{ij} - I_{ij}^{-n}). \tag{6}$$

For $D(G|n)$, by applying (4) and (3), it becomes:

$$D(G|n) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j \neq i; j=1}^{N} (I_{ij} - I_{ij}^{-n}). \tag{7}$$

A closer check on the right hand side of (6) and that of (7) reveals that they are indeed equal, because $\sum_{\forall i \neq j} (I_{ij} - I_{ij}^{-n}) = \sum_{i=1}^{N} \sum_{j \neq i; j=1}^{N} (I_{ij} - I_{ij}^{-n})$. Hence, we have

$$D(G|n) = E(G) - E(G^{-n}), \tag{8}$$

which concludes the proof. $\square$

D. The Limitation

As shown by their expressions, the three dependency impact measures $D(i \to j|n)$, $D(i|n)$ and $\Delta(G|n)$ (or equivalently $D(G|n) = E(G) - E(G^{-n})$ as discussed above) are all defined on $I_{ij} - I_{ij}^{-n}$. However, $I_{ij} - I_{ij}^{-n}$ inherently has a limitation, due to overlooking the possible fragmentation effect on the network after the deactivation of node n.

Specifically, if node j is unreachable to node i after failure or deactivation of node n, then the value of $I_{ij} - I_{ij}^{-n}$ or $D(i \to j|n)$ becomes I_{ij}, since in this case $I_{ij}^{-n} = 0$ by definition. As a result, for such cases, $D(i \to j|n) = I_{ij} - I_{ij}^{-n}$ only depends on how far node j is from node i in the presence of node n, i.e., on the value of d_{ij}, and if d_{ij} is large, $D(i \to j|n)$ will be small, giving an impression that the impact of node n on the path is small, contradicting to the fact that the path between i and j is unexistent after deactivation of n.

To demonstrate this limitation, consider a tadpole network shown in Fig. 1. In Table I, $D(1 \to j|2)$ for different j is shown, which indicates how each path starting with node 1 is impacted by the deactivation of node 2. Though simple, several surprising observations are revealed by the example.

First, as can be observed from the figure, the nodes 15 to 20 will all be unavailable to node 1 after the failure of node 2. However, Table I shows that for paths of $1 \to 15, \ldots, 1 \to 20$, their $D(1 \to j|2)$ values fall within the range $(0, 1)$ and differ from each other.

Second, $D(1 \to j|2)$ of the paths $1 \to 3$ and $1 \to 4$ are higher than of the paths $1 \to 18$ and $1 \to 19$, even though for the latter two, no path exists any more while for the former two, a path still exists after the deactivation of node 2.

Third, for $1 \to 8$ to $1 \to 14$, as easily verified from the figure, they are independent of node 2. This is reflected by their $D(1 \to j|2)$ being 0. In other words, the value 0 would naturally be considered as an indication of such independency. However, as implied by Table I, by adding m nodes sequentially to node 20, we would get for the new end node, a $D(1 \to j|2)$ value equal to $1/(7+m)$, which approaches 0 when m becomes large. In other words, the current way of calculating $D(1 \to j|2)$ gives an impression that the farther a node were from node 2, the less the node would be dependent on node 2, which is wrong.

The above observations assert that $I_{ij} - I_{ij}^{-n}$ has an evident limitation for being used as the basis in quantifying the dependency impact. Since $D(i \to j|n)$, $D(i|n)$, $D(G|n)$ and $\Delta(G|n)$ are all formulated based on $I_{ij} - I_{ij}^{-n}$, using them to index the dependency impact is void. To address this limitation, we propose a set of new measures for the purpose.

III. The Proposed Measures: Dependency Indexes

In order to take into account the fragmentation effect, the proposed measures quantify the dependency level not only based on the information measure but also on the availability of nodes. For this, a binary indicator variable A_{ij}^{-n} is used to measure the availability of node i to node j after failure of node n: $A_{ij}^{-n} = 1$ if node j is reachable to node i, and $A_{ij}^{-n} = 0$ if node j is unreachable.

The *path dependency index*, denoted as $DI(i \to j|n)$, which measures the dependency of the path $i \to j$ on node n is defined as:

$$DI(i \to j|n) \equiv \begin{cases} I_{ij} - I_{ij}^{-n} & \text{if } A_{ij}^{-n} = 1 \\ 1 & \text{if } A_{ij}^{-n} = 0. \end{cases} \tag{9}$$

There are three possible cases. One is, node j is unreachable to node i after failure of node n. In this case, the path $(i \to j)$ is totally dependent on node n and will be assigned the maximum dependency level of one. The second case is node j is reachable and there is no change in the path length. This implies that the path $(i \to j)$ is independent of node n, so $I_{ij} = I_{ij}^{-n}$ and $DI(i \to j|n) = 0$. The third case is that node j is reachable but the length of the path has increased, then the path dependency index will have a value in the range $(0, 1)$.

The *node dependency index*, denoted as $DI(i|n)$, measures the average level of dependency that node i has on node n for connecting to the other nodes, which is calculated from the path dependency index as:

$$DI(i|n) = \frac{1}{N-2} \sum_{j \in \mathcal{N}^{-n}/i \neq j} DI(i \to j|n). \tag{10}$$

There are also three possible cases. $DI(i|n) = 1$ means node i is totally dependent on node n: It is not able to connect to any other node in the network after failure of node n. $DI(i|n) = 0$ implies that node i is independent of node n, i.e. node i does not observe any connectivity change, both in terms of path length and availability. For $0 < DI(i|n) < 1$, it implies that the connectivity of node i to the rest of the nodes is affected but it can still reach at least one other node in the network.

TABLE II
Node dependency indexes: Tadpole network in Fig. 1

n \ i	1	2	6	9	15	17	19	20
2	0.38	X	0.34	0.33	0.72	0.72	0.72	0.72
9	0	0	0.02	X	0	0	0	0
15	0.27	0.27	0.27	0.27	X	0.78	0.78	0.78
19	0.05	0.05	0.05	0.05	0.05	0.05	X	1

The *network dependency index*, denoted as $DI(G|n)$, measures the average level of dependency the network G has on node n. That is $DI(G|n)$ measures the average dependency of the nodes in $\mathcal{N}^{-n}$ on node n. The network dependency index is hence calculated from the node dependency index as.

$$DI(G|n) = \frac{1}{N-1} \sum_{j \in \mathcal{N}^{-n}} DI(i|n). \tag{11}$$

Theorem 2. *If the failure of node n fragments the network G into M sub-networks where each sub-network G^m has $\alpha_m N$ number of nodes, $0 < \alpha_m < 1$, then:*

$$DI(G|n) \geq \sum_{m \in \{1..M\}} \alpha_m \Big(\sum_{k \in \{1..M\}/k \neq m} \alpha_k \Big). \tag{12}$$

Proof. A node i in a sub-network G^m will not be able to connect with the nodes in the other sub-networks after failure of node n. Let set $\mathcal{M} = \{1..M\}$. Thus,

$$DI(i|n) \geq \frac{1}{N-2} \sum_{k \in \mathcal{M} \setminus m} \alpha_k N, \tag{13}$$

where only the effect of the $\alpha_k N$ unavailable paths in each G^k, $(k \neq m)$, with $A_{ij}^{-n} = 0$ and hence $DI(i \to j|n) = 1$ is counted. Similarly, we have

$$DI(G|n) \geq \frac{1}{N-1} \sum_{m \in \mathcal{M}} \alpha_m N \Big(\frac{1}{N-2} \sum_{k \in \mathcal{M} \setminus m} \alpha_k N \Big), \tag{14}$$

which, with $\frac{N^2}{(N-1)(N-2)} \geq 1$, gives (12) and concludes. □

If all the sub-networks have equal number of nodes, i.e., $\alpha_m = \alpha$, or α is a lower bound on α_m, $\forall m$, we get

$$DI(G|n) \geq M(M-1)\alpha^2. \tag{15}$$

As a special case with $M = N$, the lower bound becomes $(N-1)/N \approx 1$ for large N. An example is a star network, where the failure of the central node makes all other nodes disconnected, so the network fully depends on the central node, i.e. $DI(G|n) = 1$, and the lower bound is approached.

Remark: As implied by their formulations, the time complexity for calculating the proposed dependency indexes is the same as for calculating the corresponding measures (3), (4) and (2) proposed in [1] and [8].

IV. Results

This section presents results of the proposed dependency indexes and an application to critical node identification.

A. Dependency Indexes

For the path dependency index, $DI(1 \to j|2)$ is shown for the tadpole network also in Table I, in comparison with $D(1 \to j|2)$. From $DI(1 \to j|2)$, we see for paths $1 \to 15$, ..., $1 \to 20$, its value is 1, implying dependency of these paths on node 2 as verified from the topology, which, however, is not reflected by $D(1 \to j|2)$ as discussed in Sec. II-D.

TABLE III
NETWORK DEPENDENCY AND RANKING OF CRITICAL NODES IN THE TADPOLE NETWORK (FIG. 1)

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$D(G\|n)$	0.022	**0.081**	0.022	0.018	0.014	0.012	0.011	0.010	0.010	0.010	0.011	0.012	0.014	0.018	**0.059**	**0.05**	**0.039**	**0.028**	0.015	0
Rank	(6)	**(1)**	(6)	(7)	(9)	(10)	(11)	(12)	(12)	(12)	(11)	(10)	(9)	(7)	**(2)**	**(3)**	**(4)**	**(5)**	(8)	(13)
$DI(G\|n)$	0.023	**0.467**	0.023	0.019	0.015	0.013	0.012	0.011	0.011	0.011	0.012	0.013	0.015	0.019	**0.41**	**0.35**	**0.28**	**0.19**	0.1	0
Rank	(7)	**(1)**	(7)	(8)	(9)	(10)	(11)	(12)	(12)	(12)	(11)	(10)	(9)	(8)	**(2)**	**(3)**	**(4)**	**(5)**	(6)	(13)
Degree	2	**3**	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1
Rank	(2)	**(1)**	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(3)

For the node dependency index, $DI(i|n)$, it is exemplified in Table II also using the tadpole network (Fig. 1). For example, the entry on column two ($i = 2$) and row three ($n = 15$) shows the value of the node dependency index $DI(2|15)$, i.e., the dependency level that node 2 has on node 15. In addition, the table shows $DI(20|19) = 1$, meaning node 20 is totally dependent on node 19, as implied by the topology.

For the network dependency index, $DI(G|n)$, it is shown and compared with $D(G|n)$ in Table III for the tadpole network. While not surprisingly, the values of $DI(G|n)$ and $D(G|n)$ are close for nodes 1, and 3 – 14, their difference is high for the other nodes. This is essentially due to the value differences in the underlying path level dependency indexes $DI(i|n)$ and $D(i|n)$ discussed above.

As a highlight, the network dependency index of node 2 is $DI(G|2) = 0.467$. Note that failure of node 2 fragments the network into two sub-networks. According to Theorem 2, the network dependency index of node two is lowered bounded by 0.44, which is very close to the actual value 0.467.

B. Application to Critical Node Identification

For the tadpole network shown in Fig. 1, Table III also compares the ranking results for critical node identification using $D(G|n)$, $DI(G|n)$ and node degree. Specifically, all these measures rank node 2 as the first, i.e. the most critical node, and node 20 the least. For the other nodes, node degree is unable to distinguish as it gives equal rank for them. $D(G|n)$ and $DI(G|n)$ place nodes 15-18 in the same order in the ranking, from 2^{nd} to 5^{th}. This ranking result is intuitive, since from the figure, their failure results in unavailability of some nodes, and the number of unavailable nodes becomes smaller in the same order. However, from node 19, the ranking becomes different. While $DI(G|n)$ still follows the same intuition and ranks node 19 at the 6^{th} place since its failure will make node 20 unavailable, $D(G|n)$ puts node 19 in the 8^{th} rank after several other nodes, which are nodes 1 and 3 (6^{th}) and nodes 4 and 14 (7^{th}) even though failure of any of these four nodes does not cause unavailability of others.

To examine the ranking difference further, a larger network as shown in Fig. 2 is considered. This network is a randomly generated scale-free network, i.e. a network with power-law degree distribution, with 500 nodes and 998 links using the Barabasi-Albert model [10]. The five most critical nodes, identified by $DI(G|n)$, are numbered from 1 to 5 corresponding to their criticality level and marked in Fig. 2. For presentation simplicity, these numbers are also used as their node numbers in the following discussion. Visually, this ranking follows the same intuition as discussed for the tadpole network, i.e. a node whose failure causes the unavailability of more nodes should be ranked higher. However, if $D(G|n)$ were used, the ranking

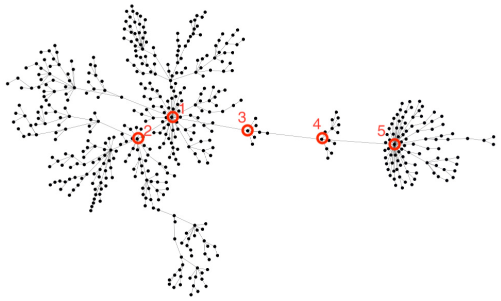


Fig. 2. Scale-free network: 500 nodes, 998 links

order among the five nodes would become 1, 2, 5, 3 and 4, even though the failure of node 5 clearly leads to unavailability of fewer nodes compared to that of node 3 or 4.

V. CONCLUSION

In this paper, the limitation of the related existing dependency measures is demonstrated and discussed. To address this limitation, a set of new measures are proposed, which assess structural dependencies at the path, node and network level. In particular, they capture fragmentation effects and hence it is possible to get, from their values, insights into the extent of fragmentation that the failure of a node will cause. The results also show that the proposed measures are better suitable for critical node identification reflecting the fragmentation effect.

VI. ACKNOWLEDGMENT

The work for this paper was performed in the context of the EU FP7 Marie Curie Actions project Grant Agreement No. 607584 (the Cleansky project).

REFERENCES

[1] V. Latora and M. Marchiori, "A measure of centrality based on network efficiency," *New Journal of Physics*, vol. 9, no. 6, p. 188, 2007.
[2] M. Lalou *et al.*, "The critical node detection problem in networks: A survey," *Computer Science Review*, vol. 28, pp. 92–117, 2018.
[3] T. N. Dinh *et al.*, "On new approaches of assessing network vulnerability: hardness and approximation," *IEEE/ACM TON*, pp. 609–619, 2012.
[4] Y. Shen *et al.*, "On the discovery of critical links and nodes for assessing network vulnerability," *IEEE/ACM TON*, vol. 21, pp. 963–973, 2013.
[5] D. F. Rueda *et al.*, "Robustness comparison of 15 real telecommunication networks: Structural and centrality measurements," *Journal of Network and Systems Management*, vol. 25, pp. 269–289, 2017.
[6] A. Bavelas, "A mathematical model for group structures," *Human organization*, vol. 7, no. 3, pp. 16–30, 1948.
[7] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and examples," *Social networks*, vol. 11, no. 1, pp. 1–37, 1989.
[8] D. Y. Kenett *et al.*, "Dependency network and node influence: Application to the study of financial markets," *International Journal of Bifurcation and Chaos*, vol. 22, no. 07, p. 1250181, 2012.
[9] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," *Physical review letters*, vol. 87, no. 19, p. 198701, 2001.
[10] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

Paper D

Towards Carrier-Grade Service Provisioning in NFV

Yordanos T. Woldeyohannes, Besmir Tola and Yuming Jiang

In Proceedings of the 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal 2019.

In the published version of paperD, in equation (2) N-1 is used in the denominator rather than N-2. However, since there are N -2 elements in $\mathcal{N}\ ^{n}/i \neq j$, for getting the value 1 in total dependency, the denominator should be N-2. Note: the results are obtained by using N - 2.

Towards Carrier-Grade Service Provisioning in NFV

Yordanos Tibebu Woldeyohannes, Besmir Tola, and Yuming Jiang
NTNU-Norwegian University of Science and Technology, Norway

***Abstract*—Network Function Virtualization (NFV) is an emerging technology that reduces cost and brings flexibility in the provisioning of services. NFV-based networks are expected to be able to provide carrier-grade services, which require high availability. One of the challenges for achieving high availability is that the commodity servers used in NFV are more error prone than the purpose-built hardware. The "de-facto" technique for fault tolerance is redundancy. However, unless planned carefully, structural dependencies among network nodes could result in correlated node unavailabilities that undermine the effect of redundancy. In this paper, we address the challenge of developing a redundancy resource allocation scheme that takes into account correlated unavailabilities caused by network structural dependencies. The proposed scheme consist of two parts. In the *first part*, we propose an algorithm to identify nodes that can be highly affected by a node failure because of their network structural dependency with this node. The algorithm analyzes such dependencies using a recently proposed centrality measure called *dependency index*. In the *second part*, a redundancy resource allocation scheme that places backup network functions on nodes considering their dependency nature and assigns the instances to flows optimally is proposed. The results show that not considering the network structural dependency in backup placement may significantly affect the service availability to flows.The results also give insights into the trade-off between cost and performance.**

I. Introduction

Middleboxes or Network Functions (NFs) are widely utilized for various purposes such as improving security and network performance. The traditional approach of having a dedicated purpose-built hardware per NF has been shown to be inefficient and expensive [1]. Network Function Virtualization (NFV) alters this inflexible architecture by decoupling the software of NFs from the hardware and run the NFs on virtualized environment such as virtual machines (VMs) or containers. NF instances can then be created on the fly depending on the traffic and the network state [2]. The VMs and containers of the NFs are usually hosted on commercial off-the-shelf (COTS) hardware, which are comparatively less expensive than the purpose-built hardware. A service in NFV is typically composed of a set of NFs that are chained in some specific order, also known as service function chaining.

Carrier-grade services such as telecommunication services require high-level of availability reaching five-nines (99.999%) or more [3] [4]. Achieving this level of availability in NFV networks is challenging due to a number of reasons including: lower dependability of COTS servers, correlated failures or unavailabilities, and state management:

COTS availability: Legacy telecommunication networks have achieved carrier-grade service availability by using purpose-built hardware. In NFV, the purpose-built hardware is replaced by COTS hardware, which is usually more error-prone [4]. To achieve the same level of availability using COTS, NFV needs to build the resilience into software [5].

Correlated failures / unavailabilities: Although most literatures assume that failures are independent, correlated failures or unavailabilities are often common in real systems [6], [7]. For example, the failure of a node may result in the unavailability of other nodes because of their structural dependencies on this node [8]. The de-facto technique for boosting availability is redundancy. However, redundancy may become ineffective due to correlated failures or unavailabilities.

State management: A large number of NFs such as Deep Packet Inspection (DPI) and Network Address Translator (NAT) are stateful. Stateful NFs preserve service states, such as, TCP connection state and the mapping between IP addresses about ongoing connections [9]. Typically, NFs need to maintain 10-100s of state variables that are per-flow or shared across flows [10]. Backup instances of stateful NFs need to have updated state information to ensure successful failover and service continuity [11], [12].

In this paper, we make a step forward towards carrier-grade service provisioning in NFV, by proposing a novel redundancy resource allocation scheme where two crucial challenges are addressed. (1) One challenge is *how to factor the inherent network structural dependency among nodes into redundancy resource allocation*. (2) The other challenge is *how to efficiently place and allocate backup instances for service chain of flows.*

To tackle the first challenge, an algorithm that measures the dependency among network nodes and identifies nodes that have a high-level of structural correlation is proposed. The dependency among nodes of a network is quantified by using a centrality measure called *node dependency index* [13]. Here, there is an intuition, which is, a backup NF should not be placed at a node that may also become unavailable when the primary NF's node fails. For the second challenge, an optimization model that aims to efficiently place redundant NFs and assign backup NFs to service chains of flows is proposed. In this model, flows are assigned backup instances following the $1 : m$ active-standby redundancy mode, with which, every flow can tolerate the failure of any one of the NF instances on the NF chain [11]. In addition, following the intuition, redundant instances are not placed on backup nodes that are structurally correlated with the primary nodes. Furthermore, to efficiently utilize resources, backup NFs are shared by flows and only the required number of instances are created.

Most of the existing works on redundancy allocation in NFV based networks focus on providing two-nines or three-nines service availability [14], [15]. only a few consider carrier-grade service availability [16], [17]. Both [16] and [17] allocate only on-site redundancies. However, to guarantee carrier-grade service availability it is important to also have backups distributed geographically [4]. In addition, [17] considers only the failure of the physical nodes while not the NF applications and assumes that all nodes have the same availability, and [16] focuses on the failure of VMs assuming similar availability and failure independence between VMs.

Our proposed scheme differs from the existing works in a number of ways. First, our scheme considers the effect of network structural dependency. Second, it takes into account both physical hardware failures and NF software failures with different availability values. Third, it can be used to allocate both on-site and off-site backups in an optimal way. Moreover, in our proposed scheme, in addition to availability, delay performance is also taken into consideration, such that the delay that flows experience after failover can be kept within the requirement of the flows.

The specific contributions in this paper include:

- An algorithm that identifies the set of nodes that have strong structural correlation using a centrality measure called node dependency index.
- A redundancy allocation scheme that finds the optimal number and placement of backup nodes and NF instances and assigns the instances to flows.

The paper is organized as follows. In Section II, the system model is described. Section III discusses in brief the node dependency index centrality measure. In Section IV, the algorithm proposed for identifying the nodes that have high-level of structural correlation is explained. The proposed redundancy allocation scheme is presented in Section V, followed by the results in Section VI. Finally, Section VII presents the concluding remarks.

II. System Model

The system considered is a network of nodes and links and is represented as a graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{L}$ represents the set of links. Nodes hosting NFs that are being utilized by the primary service chains of flows are called *primary nodes*. Nodes that can be used for backup are called *backup nodes*. $\mathcal{B}$ denotes the set of backup nodes and $\mathcal{P}$ the set of primary nodes. Backup NFs are hosted on backup nodes.

A node hosting backup instances can be a shared or dedicated backup node. A shared backup node is a node that is being used both as primary and backup node. This type of nodes reserve some resources to be used as backup while hosting NFs that are utilized by the primary service chains. Dedicated backup nodes are nodes that are used only to host backup instances.

Each flow f is defined by a source and destination node pair, which are respectively denoted as s_f and d_f. The service required by flow f is represented by a service chain, $\overrightarrow{S}_f = (S_f^1, S_f^2 \dots S_f^{g_f})$. The service chain is an ordered series of network functions, where S_f^1 is the first NF, S_f^2 is the second NF needed and so on. It is assumed that a flow is already assigned a primary service chain. The variable $p_{f,g}$ indicates the primary node p that is serving flow f's g^{th} service. A backup instance of an NF of type v requires k_v number of cores and can be a backup to up to C_v^b number of flows. For each flow f, there is an availability requirement on its service $\overrightarrow{S}_f$, denoted as A_f. A service $\overrightarrow{S}_f$ is considered available is either the primary or one of the backup service chains is available.

III. Structural Dependency Measures

The *node dependency index* $DI(i|n)$ measures the average level of dependency node i has on node n in connecting with the other nodes of the network [13]. $DI(i|n)$ is calculated from the path dependency index $DI(i \to j|n)$, which measures the dependency the path between nodes i and j has on node n. $DI(i \to j|n)$ is defined as:

$$DI(i \to j|n) \equiv \begin{cases} I_{ij} - I_{ij}^{-n} & \text{if } A_{ij}^{-n} = 1 \\ 1 & \text{if } A_{ij}^{-n} = 0, \end{cases} \tag{1}$$

where I_{ij} is an information measure, which is equal to the inverse of the shortest path distance hop counts, denoted as d_{ij}, between nodes i and j, i.e. $I_{ij} = 1/d_{ij}$. I_{ij}^{-n} is the information measure between nodes i and j after the deactivation of node n. The binary variable A_{ij}^{-n} measures the availability: $A_{ij}^{-n} = 1$ if node i can reach node j after the deactivation of node n and zero otherwise. The node dependency index is defined as:

$$DI(i|n) = \frac{1}{N-2} \sum_{j \in \mathcal{N}^{-n}/i \neq j} DI(i \to j|n). \tag{2}$$

$DI(i|n)$ measures the average dependency that node i has on node n. $DI(i|n) = 1$ if node i cannot connect with the other nodes, $DI(i|n) = 0$ if node i does not experience any connectivity problem and $0 < DI(i|n) < 1$, if node i experiences connectivity problem but is still able to connect to at least one other node, all after the failure of node n.

IV. Structurally Correlated Nodes

While failure independence is commonly assumed when studying availability, recent studies have demonstrated the existence of correlated failures and the pronounced effect of geographical adjacency [7] [18], [19]. Nevertheless, it has also been recognized that it is difficult to discover or predict dependencies among failures [6], [7] [18], [19]. To tackle this challenge, in this section, a novel approach is proposed to identify nodes that are inherently correlated due to the network structure. This information lays a foundation for the proposed redundancy allocation scheme that will be detailed in Section V.

A. Algorithm

The failure of a node may result in the unavailability of other nodes. For example, in a data-center network, the failure of a Top-of-Rack switch will result in the unavailability of all the servers located in the same rack. The proposed algorithm uses the node dependency index to measure the dependency among nodes and identify the nodes that have high structural correlation. From the definition of the node dependency index, if node i has high-level of dependency on node n, the failure of node n might result in the unavailability of node i.

Definition 1: *Critical nodes of node i*, denoted as $\mathcal{C}(i)$, are nodes that node i is highly dependent on, where node i is said to be highly dependent on node n if $DI(i|n)$ is above a given threshold t_{DI},

$$\mathcal{C}(i) = \{n|DI(i|n) > t_{DI}, n \in \mathcal{N}\}. \tag{3}$$

If $\mathcal{C}(i)$ is empty, node i is independent of the other nodes of the network so has no critical node. For example, in a fully mesh network, all nodes are independent of each other as the failure of one does not affect the connectivity of the others.

To find the set of nodes that have strong structural correlation with a primary node i, some intuitive observations are used which include:

First-level dependency

- Node i has a high probability of experiencing a correlated failure with its critical nodes in $\mathcal{C}(i)$ as the failure of these nodes might lead to the unavailability of node i. Thus, node i should not use the nodes in $\mathcal{C}(i)$ as a backup.
- Node i should not also use as a backup those nodes that depend on it highly. Since the failure of node i might also result in the unavailability of these nodes.

In brief, *a primary node i and its backup nodes should not depend on each other.* This can be regarded as the *first-level dependency* among nodes.

Second-level dependency

- Node i should not use as a backup nodes that depend heavily on its critical node. This is because if the unavailability of node i is due to the failure of its critical node, the other nodes that depend heavily on the critical node might also be unavailable.

The algorithm for finding a set of nodes, $\hat{\mathcal{B}}_i$, which are structurally correlated with a primary node i is shown in Algorithm 1. The algorithm starts by finding the critical nodes of a primary node. The critical nodes of node i, $\mathcal{C}(i)$, are inserted into the set $\hat{\mathcal{B}}_i$. Then, nodes that have a high-level of dependency on node i are included to the set $\hat{\mathcal{B}}_i$ (line 9). For the second-level dependency, all the nodes that are highly dependent on the critical nodes of node i will be included to $\hat{\mathcal{B}}_i$. The threshold, t_{DI}, should be assigned values that are between zero and one. If it is set to a very low value that is close to zero, the set $\mathcal{C}(i)$ will include a large number of network nodes. Therefore, it should be assigned a relatively large or medium values such as 0.5.

Algorithm 1 Heuristic for finding structurally correlated nodes

```
1:  $G(\mathcal{N},\mathcal{L}) \rightarrow$ the network graph.
2:  $\hat{\mathcal{B}}_i \rightarrow$ set of nodes that are structurally correlated with node $i$.
3:  $t_{DI} \rightarrow$ threshold for high dependency
4:  for $i \in \mathcal{N}$ do
5:      Find $\mathcal{C}(i)$ using $t_{DI}$
6:      Insert $\mathcal{C}(i)$ to $\hat{\mathcal{B}}_i$
7:      for $j \in \mathcal{N} / i$ do
8:          if $i \in \mathcal{C}(j)$ then
9:              Insert $j$ to $\hat{\mathcal{B}}_i$
10:         if $j \in \mathcal{C}(i)$ then
11:             for $k \in \mathcal{N} / i$ do
12:                 if $j \in \mathcal{C}(k)$ then
13:                     Insert $k$ to $\hat{\mathcal{B}}_i$
14: return $\hat{\mathcal{B}}_i$
```

V. Redundancy Allocation Scheme

Some of the features considered in the design of the redundancy allocation scheme include:

- *Correlated failures*: To tolerate correlated failures caused by network structural dependency, backup NFs of a flow are not placed on nodes that are structurally correlated with the primary nodes of the flow.
- *State:* Stateful NFs can have states that are per-flow or shared across flows. Flows using the same primary stateful NF instance will be assigned to the same backup instance since they rely on a state shared among them.
- *Delay vs utilization:* To efficiently use network resources, minimal number of backup instances are created. However, this can increase the end-to-end backup chain delay of flows. To solve this problem, the scheme finds a balance between minimizing the backup chain delay, which is the delay flows will experience after failover, and the resource utilization.

A. Formulation: All-One

The first model considered is the All-One model in which *all* the services of a chain are assigned one backup that is a 1:1 active-standby mode. It is assumed that *one* backup node will be used to backup all the NFs of a flow, later on this assumption will be relaxed. The redundancy allocation is then formulated as an Integer Linear Program (ILP).

The redundancy allocation has three main objectives: (I) *to minimize the number of backup instances created*, (equation (4)), (II) *to minimize the number of backup nodes used*, (equation (5)), and (III) *to minimize the backup chain delay*, (equation (6)).

$$\text{minimize} \quad \sum_{\forall b \forall v} z_v^b \tag{4}$$

$$\text{minimize} \quad \sum_{\forall b} q^b \tag{5}$$

$$\text{minimize} \quad \sum_{\forall b \forall f} (D(s_f, b) i_f^b + D(b, d_f) i_f^b) \tag{6}$$

The weighted sum method is used to combine the three objective functions into one by using equal unit weights. For positive weights, the optimal solution of the single-objective representation is also a Pareto optimal solution of the multi-objective problem [20]. The All-One optimization model is given us:

All-One:

$$\text{min.} \sum_{\forall b \forall v} z_v^b + \sum_{\forall b} q^b + \sum_{\forall b \forall f} (D(s_f, b) i_f^b + D(b, d_f) i_f^b). \quad (7)$$

s.t.

$$1 - (1 - \sum_{\forall b} i_f^b * A^b \prod_{g=1, v=S_f^g}^{g_f} A_v)(1 - A_{s_f}^p) \geq A_f \quad , \forall f : A_{s_f}^p < A_f \quad (8a)$$

$$\sum_{\forall f, \forall g / S_f^g = v} y_{f,g}^b \leq C_v^b \quad , \forall b, v \quad (8b)$$

$$\sum_{\forall v} z_v^b * k_v \leq K^b \quad , \forall b \quad (8c)$$

$$y_{f,g}^b = 0, \quad , \forall f, g \in \{1..g_f\}, \forall b \in \hat{\mathcal{B}}_p / p \in \mathcal{P}_f \quad (8d)$$

$$y_{f,g}^b = y_{f',g'}^b \quad , \forall f, f' \in \mathcal{F} / p_{f,g} = p_{f',g'}, \; S_f^g = S_{f'}^{g'}, T(S_f^g) = 1, \forall b, g, g' \quad (8e)$$

$$\sum_{\forall b} y_{f,g}^b = 1 \quad , \forall f : A_{s_f}^p < A_f, g \in \{1..g_f\} \quad (8f)$$

$$\sum_{\forall b} i_f^b = 1 \quad , \forall f : A_{s_f}^p < A_f \quad (8g)$$

$$y_{f,g}^b \leq z_v^b \quad , \forall b, f, g \in \{1..g_f\}, v = S_f^g \quad (8h)$$

$$q^b = \max_{v \in \mathcal{V}} z_v^b \quad , \forall b \quad (8i)$$

$$i_f^b = y_{f,g}^b \quad , \forall f, b, g \in \{1..g_f\} \quad (8j)$$

The constraints are classified into five group, which are *availability*, *capacity*, *correlated failure*, *state* and *assignment* constraints. Constraint (8a) belongs to the *availability* group and ensures that flows' availability requirement is fulfilled by the primary and backup NFs assigned. Constraints (8b) and (8c) are *capacity* constraints for the backup NF instances and backup nodes respectively. For each flow, constraint (8d) prohibits the usage of backup nodes that have high structural *correlation* with the primary nodes of the flow. Constraint (8e) is a *state* constraint, which makes sure that flows using the same primary instance of a stateful NF are assigned to the same backup instance.

The other constraints belong to the *assignment* group. *All* the services of a flow have a backup (constraint (8f)) and only *one* backup node hosts all the backup NFs of a flow (constraint (8g)). A flow is mapped to a backup instance on a given backup node only if the node is hosting the NF type (Constraint (8h)). Constraint (8i) identifies backup nodes that are hosting instances. A flow is assigned to a backup node only if it is using backup instance hosted on the node (Constraint (8j)).

TABLE I: Symbols used in formulation

Notation	**Meaning**
K^b	the number of cores available on backup node b.
$p_{f,g}$	primary node p is used by flow f's g^{th} service.
$D(p, b)$	the delay between nodes p and b.
A^b	the probability that backup node b is available.
$\hat{\mathcal{B}}_p$	set of backup nodes that have high structural correlation with node p.
k_v	the number of cores needed to instantiate NF type v.
$T(v)$	binary variable to show if NF type v is stateful ($T(v) = 1$) or not ($T(v) = 0$).
C_v^b	the maximum number of flows that an instance of NF type v hosted on node b can be a backup to.
A_v	the probability that the application software of a network function of type v is available.
A_f	the availability requirement of flow f.
s_f, d_f	source and destination nodes of flow f respectively.
$\overrightarrow{S}_f = (S_f^1, S_f^2 \dots S_f^{g_f})$	service chain of flow f.
A_{sf}^p	the availability of the primary service chain of flow f.
$\mathcal{P}_f$	the set of primary nodes used by flow f.
Decision variables	
$y_{f,g}^b$	a binary decision variable, to indicate if backup node b is used as a backup for flow f's g^{th} service.
z_v^b	an integer decision variable to indicate the number of backup instances of NF type v hosted on backup node b.
i_f^b	a binary variable that indicates if backup node b is used by flow f or not.
q^b	a binary variable to indicate if backup node b is hosting backup NF instances.

B. Formulation: All-Any

The "All-One" ILP model given above uses one backup node to backup all the NFs of a flow. This constraint is relaxed so that a flow can use one or more backup nodes. This model will be referred as "All-Any" since *all* of the services of a flow are backed up and a flow can use *any* number of backup nodes. The objective function for minimizing the backup chain delay needs to be modified as

All-Any:

$$\text{minimize} \sum_{\forall b \forall f} (D(s_f, b) y_{f,1}^b + D(b, d_f) y_{f,g_f}^b + \sum_{\forall b' \in \mathcal{B}, g=1}^{g_f - 1} D(b, b') y_{f,g}^b * y_{f,g+1}^{b'}). \quad (9)$$

Since a flow might use more than one backup node, the backup chain delay will include the delay between the backup nodes. All the constraints except three (8a, 8g and 8j) of the All-One model will also be included in the All-Any model. The three constraints will be replaced by constraints (10, 11 and 12) respectively.

$$1-(1-\prod_{\forall b}\max(1-i_f^b, i_f^b A^b\prod_{g=1}^{g_k}\max(1-y_{f,g}^b, y_{f,g}^b A_v)))$$
$$(1-A_{s_f}^p)\geq A_f \qquad \forall f: A_{s_f}^p < A_f \quad (10)$$

$$\sum_{\forall b} i_f^b \geq 1 \qquad \forall f: A_{s_f}^p < A_f \quad (11)$$

$$i_f^b = \max_g(y_{f,g}^b) \qquad \forall b, \forall f: A_{s_f}^p < A_f \quad (12)$$

Constraint (10) guarantees that the availability requirement of flows is satisfied by the primary and backup instances, which might be hosted on different backup nodes. One or more backup nodes are assigned to a flow (constraints (11)). A flow is assigned a backup node provided that it is using atleast one backup instance hosted on the node (constraint (12)). This model is an Integer Non-linear Program (INLP) because of the non linearity of equation (10). To decrease the complexity of the model, the non-linear constraint is approximated by a linear equation.

1) Linear approximation: The availability constraint is approximated by a linear lower bound function.

Theorem 1. *The probability that all E entities of a set $\mathcal{E}$ will be available is lower bounded by $1-\sum_{e=1}^{E}U_e$, where every entity $e\in\mathcal{E}$ fails independently with probability A_e and U_e is the unavailability of entity e.*

Proof: The probability that all the E entities will be available (A_t) is a product of the availability of each of them. That is

$$A_t=\prod_{e=1}^{E}A_e. \quad (13)$$

By definition, the availability of entity e, $A_e = 1-U_e$, where U_e is the unavailability of e. Thus,

$$A_t=\prod_{e=1}^{E}(1-U_e). \quad (14)$$

By expanding the product,

$$\prod_{e=1}^{E}(1-U_e)=1-\sum_{e=1}^{E}U_e+\sum_{e=1}^{E-1}U_e*U_{e+1}+o(n), \quad (15)$$

where $o(n)$ represents the higher order terms. Usually, unavailability $U<<1$ so the product and the higher order terms can be ignored. We will then have

$$\prod_{e=1}^{E}(1-U_e)\geq 1-\sum_{i=1}^{E}U_e. \quad (16)$$

As a result,

$$A_t\geq 1-\sum_{e=1}^{E}U_e, \quad (17)$$

which concludes the proof. ■

For example, if there are two entities with availability $A_1 = 0.9$ and $A_2 = 0.99$, then $A_t = 0.891$. Using the linear approximation, $U_1 = 0.1, U_2 = 0.01$, so $A_t = 0.89$. Thus, the linear approximation is a lower bound to the actual availability value. Applying this linear approximation, equation (10) can be approximated by:

$$1-(1-(1-\sum_{\forall b}i_f^b(U^b+\sum_{g=1}^{g_k}y_{f,g}^bU_v)))(1-A_{s_f}^p)$$
$$\geq A_f \qquad ,\forall f: A_{s_f}^p < A_f, \quad (18)$$

where U^b and U_v are the unavailabilities of backup node b and backup NF v respectively. The lower bound approximation is conservative so flows availability requirement will not be violated. Equation (18) is not linear since it has a term that is a product of two variables, i_f^b and $y_{f,g}^b$. Let variable $r_{f,g}^b = i_f^b * y_{f,g}^b$,

$$1-(1-(1-\sum_{\forall b}i_f^bU^b-\sum_{\forall b}\sum_{g=1}^{g_k}r_{f,g}^bU_v))(1-A_{s_f}^p)$$
$$\geq A_f \qquad ,\forall f: A_{s_f}^p < A_f. \quad (19)$$

Equation (19) is a linear equation of the variables i_f^b and $r_{f,g}^b$. Thus, the non-linear inequality constraint in equation (10), can be substituted by equation (19) and the constraint $r_{f,g}^b = i_f^b * y_{f,g}^b$. However, since the variables i_f^b and $y_{f,g}^b$ are binary, their product can easily be linearized by substituting it with the following linear equations,

$$r_{f,g}^b <= i_f^b$$
$$r_{f,g}^b <= y_{f,g}^b$$
$$r_{f,g}^b >= i_f^b + y_{f,g}^b - 1. \quad (20)$$

Thus, the equivalent ILP model of the All-Any model will have constraints (19) and (20) instead of the non-linear availability constraint and all the other linear constraints of the All-Any INLP model.

C. Allocating more than one backup chain

The All-One and All-Any models assign one backup for each of the NFs of a flow's service chain. However, to guarantee the high availability of carrier-grade services, it might be necessary to allocate more than one backup chain. The following simple example is used to show case this. Consider a flow that has two services long chain. The primary chain of the flow is 90% available and the flow requires to be 99.999% available. The backup nodes and NF applications are 99% and 99.9% available respectively. Thus, after being allocated backup instances that are hosted on the same backup node, the flow will only be 99.88% (2'9s) available. Thus, a second backup chain is needed to reach the required 99.999% (5'9s) availability.

Algorithm for assigning more than one backup chain: Backup instances are to be allocated for a set ($\mathcal{F}$) of flows. The proposed models assign one backup chain. More than one backup chains are allocated to a flow sequentially one after the other. That is the first backup chain is allocated and if the availability requirement of the flow is not satisfied then

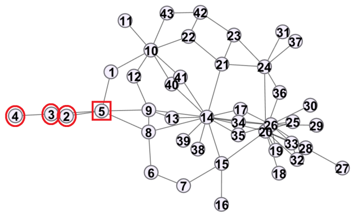


Fig. 1: GEANT network: Example of structurally correlated nodes

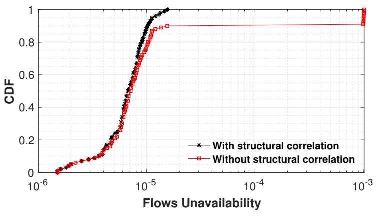


Fig. 2: Effect of not considering structural correlation: CDF of the unavailability

the second backup chain is assigned and so on. One problem with using the models directly for assigning backup chains sequentially is that if the availability requirement of a flow $f \in \mathcal{F}$ cannot be satisfied by one backup chain, the models will be infeasible. To solve this issue, for all of the flows in the set, it is checked whether their availability requirement can be satisfied while being assigned to the least available backup node. If not, the availability requirement of the flow will be downgraded to the next availability class, e.g., from 99.999% to 99.99% or from 99.99% to 99.9%. The original availability requirement of the flow is saved and the flow will be marked as a flow that might need more than one backup.

Then, the first backup chain will be allocated by using the models. The state of the network (including the placement of backup instances and their capacity) will then be updated. If the availability requirement of a flow is not satisfied by the first backup chain, then the same process will be used to allocate the second backup chain. Two variables are introduced to transfer the state of the network between the different rounds of backup chain allocations. These are oZ_v^b, the number of backup instances of type v already created on node b, and oC_v^b, the currently available capacity of an existing instance of type v hosted on node b. For this algorithm, in the models z_v^b will be replaced by $z_v^b + oZ_v^b$ and C_v^b will be replaced by $C_v^b + oC_v^b$. This is done to be able to use instances created previously in the current round of the backup allocation.

In case it is not possible to allocate backups due to shortage of resources, flows will be rejected. Resource shortage can occur at any round of the backup chains assignment. For example, when a flow is allocated a second backup chain. In this case, the flow has already been assigned one backup chain. However, the availability requirement of the flow is not yet satisfied. In cases like this, the flow will be rejected and the resources already assigned to it will be released to be used by other flows.

VI. Results

The performance of the proposed scheme is analyzed by conducting a number of experiments. The models are solved by using a commercial solver, CPLEX, together with Matlab for transferring updated network state information between different rounds. The GEANT network, Fig. 1, which is the pan-European research and education network, is used as a test network topology [21].

Eight nodes of the network are chosen to be the ingress/egress nodes. The ingress/egress nodes are a bottleneck for achieving high availability since the failure of one of them leads to service unavailability for customers using it. To avoid this, these nodes are paired to provide "dual homing", whereby one node is a backup for the other and vice versa. Twenty-two nodes of the network are assumed to be backup nodes (in shared or dedicated mode). Each backup node has 4 CPU cores to be used by the backup instances it hosts. The rest of the nodes are dedicated primary nodes. The availability of the nodes is assumed to be uniformly distributed between $0.99 - 0.999$, whereas NF instances have an availability between $0.999 - 0.9999$ and an NF instance can be a backup for up to 10 flows.

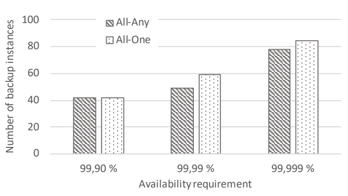
Flows are assumed to require a service chain that is composed of two NFs. NFs of a chain are randomly chosen out of the set of five services, which are Firewall, DPI, IDS, Proxy, NAT. Flows are assigned primary chains by using ClusPR algorithm [2]. The availability requirement of a flow is selected from the set $\{0.999, 0.9999, 0.99999\}$.

A. Structurally correlated nodes

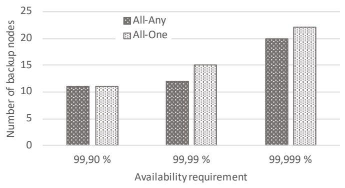
Example of nodes that have high structural correlation, which are identified by the proposed algorithm are highlighted in Fig. 1. Nodes 2-4 have a high probability of experiencing a correlated failure with node 5 due to their dependencies. This is because, the failure of node 5 will lead to the unavailability of these nodes as well.

1) Effect of not considering structural correlation: In this section, a simple experiment is carried out to showcase the effect of not considering the structural correlation among nodes in the backup instance placement decision making. The baseline algorithm from [16] is used to decide the number of backup instances needed. It is assumed that the availability of a node is 0.999 (3'9s). According to the baseline algorithm, theoretically, one backup for each of the NFs is enough to meet the 99.999% availability requirement of a single service function chain containing two NFs. The primary and backup NF host nodes of a chain are randomly chosen from the network. When structural correlation is considered, the backup nodes of a chain will not have strong correlation with the primary nodes.

The availability of 100 flows is measured by conducting ten million simulation runs. In each simulation run, the state of each node, i.e., failed (0) or up (1), is randomly generated

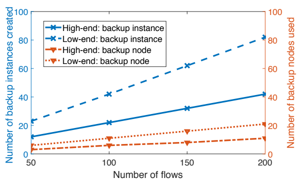


(a) Backup instances

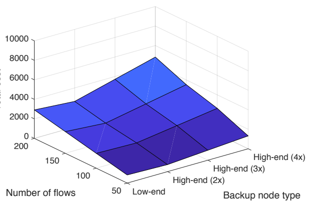


(b) Backup nodes

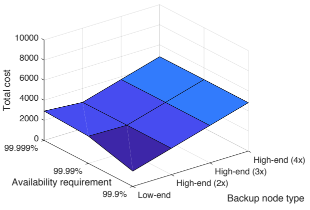
Fig. 3: Number of backup NF instances and nodes utilized for acheiving different availability requirements



(a) Backup nodes and instances



(b) Cost for achieving 99.999% availability



(c) Cost for 200 flows

Fig. 4: “High-end” vs “Low-end”: Number of backup NF instances created and backup nodes used (a), total cost for achieving 5’9s (b), total cost for 200 flows (c).

from Bernoulli distribution using the node’s availability. The network is then updated considering the nodes state. Finally, the availability of the backup and primary chains is checked by verifying the availability of the host nodes of the chain’s NFs and the path between them. The chain is said to available if either the primary or backup chain is available. A CDF of the unavailability of the 100 flows is shown in Fig. 2. When structural correlation is not considered around 10% of the flows have low availability, 3’9s and 2’9s.

B. Resource utilization

The number of backup instances created and nodes utilized for fulfilling the availability requirements of 200 flows for different availability requirements are shown in Fig. 3a and 3b respectively.

When the flows have 99.9% availability requirement, 42 backup instances are created and 11 backup nodes are used to host the instances by both All-Any and All-One models. The availability requirement of all of the flows is able to be fulfilled with 1:1 active-standby backup for each of the NFs of a chain. When the availability requirement of the flows increases to 99.99%, 49 backup instances are created by the All-Any model and 59 by the All-One model. For some of the flows, 1:1 active-standby was not enough to reach to the required availability therefore, a 1:2 active-standby, where one NF of a chain has two backup instances, is required. As a result, more backup instances are created. When comparing the two models, the All-One model created more instances than the All-Any model. This is because of the All-One model’s constraint that forces a flow to use backup instances hosted only on the same node. For achieving 5’9s (99.999%) availability, most of the flows need 1:2 backup protection.

C. Effect of the type of backup nodes

In this section, the effect of using highly available COTS servers versus COTS with lower availability, referred to as “High-end” and “Low-end” respectively is analyzed. The “High-end” servers are assumed to be 99.9% available and the “Low-end’ servers 99% available. The relative importance between the cost of installation of host server hardware and the cost of installation of the network function software license is chosen to be 100:10 for “Low-end” servers as in [22]. For the “High-end” servers, the cost is 200:10 if the “High-end” servers are twice (2×) more expensive, and 300:10 if they are 3× more expensive compared to the “Low-end”.

Figure 4a and 4b show the number of backup instances created, nodes utilized and the total cost for fulfilling 99.999% availability requirement of different number of flows. For the “Low-end” servers, 1:2 backup have to be applied to reach the 5’9s requirement. Using the “High-end” servers, the availability requirement is fulfilled with 1:1 active-standby redundancy. Therefore, fewer backup instances are created when using “High-end” servers. However, the “High-end” servers are more expensive than the “Low-end” servers. The total cost spent for fulfilling the availability requirement of the flows depends on the relative cost of the servers. It is more economical to use “High-end” servers if their cost is not more than 2× the cost of the “Low-end” servers. If the cost of installation of the “High-end” servers is 3× or more compared to the “Low-end” servers, the total cost spent will be more than that spent using the “Low-end” servers.

Figure 4c shows the total cost for serving 200 flows when their availability requirement changes. The 1:1 active-standby protection is enough for meeting the 99.9% avail-

TABLE II: Effect of including delay in the objective function

Objective	**Average delay (hops)**	**Worst-case delay (hops)**	**# Backup instances**
Without delay	4.68	12	12
With delay	0.44	2	15

ability requirement. Thus, it is economical to use the "Low-end" servers. For both 99.99% and 99.999%, if the "High-end" servers are 2× more expensive or less, then it is more economical to use the "High-end" servers.

D. Effect of minimizing the backup chain delay

The backup chain delay, in terms of number of hops, observed when the objective includes minimizing the end-to-end delay and the amount of resources used is compared with the case when the objective is only to minimize the resources used (i.e., instances created and nodes utilized).

Table II shows the results of the comparison for 50 flows that have 99.9% availability requirement. When the objective is to minimize the resource utilization, 12 instances are created. Compared to the primary chain, the backup chain delay is 4.68 hops longer on average. In the worst case, a flow's backup chain is 12 hops longer than its primary one. When the objective function is to minimize both the total backup chain delay and the resource utilization, the average backup chain delay is only 0.44 hop counts longer and the worst-case observed delay is 2 hops longer. In this case, 15 backup instances are created.

VII. Conclusion

In this paper, a redundancy resource allocation scheme that tolerates correlated failures caused by network structural dependency is proposed. The scheme identifies the sets of nodes that have strong structural correlation using a novel algorithm that is based on the node dependency index centrality measure. The experimental results demonstrate that not taking into account the structural correlation among nodes in backup instances placement decision making considerably affects the availability of flows. The results also give insights into the trade-off between cost and system performance.

References

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "ClusPR: Balancing multiple objectives at scale for NFV resource allocation," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2018.

[3] D. Collins, *Carrier grade voice over IP*. McGraw-Hill New York, 2003, vol. 2.

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

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[6] H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz, "Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems," in *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*. IEEE, 2002, pp. 362–367.

[7] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Subtleties in tolerating correlated failures in wide-area storage systems." in *NSDI*, vol. 6, 2006, pp. 225–238.

[8] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, "Heading off correlated failures through independence-as-a-service." in *OSDI*, 2014, pp. 317–334.

[9] S. Woo, J. Sherry, S. Han, S. Moon, S. Ratnasamy, and S. Shenker, "Elastic scaling of stateful network functions," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 299–312.

[10] J. Khalid, A. Alsudais, E. Keller, and F. Le, "Paving the way for NFV: Simplifying middlebox modifications using statealyzr." in *NSDI*, 2016, pp. 239–253.

[11] N. ISG, "Network function virtualisation (NFV)-resiliency requirements," *ETSI GS NFV-REL*, vol. 1, p. v3, 2016.

[12] P. X. Sherry, Justine Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. Martins, S. Ratnasamy, L. Rizzo *et al.*, "Rollback-recovery for middleboxes," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 227–240.

[13] Y. T. Woldeyohannes and Y. Jiang, "Measures for network structural dependency analysis," *IEEE Communications Letters*, 2018.

[14] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[15] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[16] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*. IEEE, 2017, pp. 1–10.

[17] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Globecom Workshops (GC Wkshps), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[18] A. J. Gonzalez, B. E. Helvik, J. K. Hellan, and P. Kuusela, "Analysis of dependencies between failures in the UNINETT IP backbone network," in *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*. IEEE, 2010, pp. 149–156.

[19] B. Chun and A. Vahdat, "Workload and failure characterization on a large-scale federated testbed," *Intel Research Berkeley, Tech. Rep. IRB-TR-03-040*, 2003.

[20] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[21] *GEANT the pan-european research and education network,*, 2018 (accessed Septemper 10, 2018). [Online]. Available: http://www.geant.net.

[22] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in nfv deployments," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

Paper E

CoShare: An Efficient Approach to Redundancy Allocation in NFV

Yordanos T. Woldeyohannes, Besmir Tola, Yuming Jiang, and K. K. Ramakrishnan

Available on arXiv at https://arxiv.org/abs/2008.13453, and to be submitted to a journal.

This article is awaiting publication and is therefore not included.

ISBN 978-82-326-5146-7 (printed ver.)
ISBN 978-82-326-5147-4 (electronic ver.)
ISSN 2703-8084 (online)
ISSN 1503-8181 (printed ver.)

NTNU
Norwegian University of
Science and Technology