On the Design of a Cost-efficient Resource Management Framework for Low Latency Applications

Binxu Yang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Electronic and Electrical Engineering

University College London

July 23, 2018

I, Binxu Yang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The ability to offer low latency communications is one of the critical design requirements for the upcoming 5G era. The current practice for achieving low latency is to overprovision network resources (e.g., bandwidth and computing resources). However, this approach is not cost-efficient, and cannot be applied in large-scale. To solve this, more cost-efficient resource management is required to dynamically and efficiently exploit network resources to guarantee low latencies. The advent of network virtualization provides novel opportunities in achieving cost-efficient low latency communications. It decouples network resources from physical machines through virtualization, and groups resources in the form of virtual machines (VMs). By doing so, network resources can be flexibly increased at any network locations through VM auto-scaling to alleviate network delays due to lack of resources. At the same time, the operational cost can be largely reduced by shutting down low-utilized VMs (e.g., energy saving). Also, network virtualization enables the emerging concept of mobile edge-computing, whereby VMs can be utilized to host low latency applications at the network edge to shorten communication latency. Despite these advantages provided by virtualization, a key challenge is the optimal resource management of different physical and virtual resources for low latency communications.

This thesis addresses the challenge by deploying a novel cost-efficient resource management framework that aims to solve the cost-efficient design of 1) low latency communication infrastructures; 2) dynamic resource management for low latency applications; and 3) fault-tolerant resource management.

Compared to the current practices, the proposed framework achieves 80% of deployment cost reduction for the design of low latency communication infrastructures; continuously saves up to 33% of operational cost through dynamic resource management while always achieving low latencies; and succeeds in providing fault tolerance to low latency communications with a guaranteed operational cost.

Acknowledgements

First of all, I would like to thank my supervisor, Professor George Pavlou, for giving me the opportunity to pursue a Ph.D., and for constantly providing me with his mentoring, advice, encouragement and support. Also, I would like to thank Dr. Wei Koong Chai for his countless support, advice and insightful discussions, which made this thesis possible. Most importantly, I would like to thank both of them for teaching me not only how to do research, but also how to solve every complex problem with critical thinking, which I can benefit from for the rest of my life.

I also benefited from the advice and help provided by many mentors that I encountered along my Ph.D. In particular, I would like to thank Dr. Zichuan Xu and Dr. Daphne Tuncer, who guided me into the world of optimization, and provided me insightful comments for different research questions tackled in this thesis. In addition, I would like to thank Dr. Konstantinos V. Katsaros for his tremendous help in the initial stage of my Ph.D. Last, I would also like to thank Dr. Truong Khoa Phan, Dr. Onur Ascigil, and all the colleagues of the NSRL group for the encouragement and advice throughout my Ph.D.

I would also like to thank the external and internal examiners of my Ph.D., Professor Kun Yang (University of Essex) and Professor Miguel Rio, for all their valuable comments and feedback concerning this thesis.

Finally, I would like to thank my family and friends. In particular, I am deeply grateful to my parent, Hong Yang and Dr. Yanling Sun. They provided me constant encouragement, support and endless love throughout my life, which has been a major source of inspiration for all my achievements.

Contents

List of Figures

List of Tables

Abbreviations

AB-PLeC	Application-Level Betweeness and Path Length Constraint
ADN	Active Distribution Network
ALB	Auto-Scaling and Load Balancing
AP	Access Point
AR	Augmented Reality
CAPEX	Capital Expenditure
CSCP	Capacitated Set Covering Problem
CVD	Capacity Violation Detection
DC	Data Center
DER	Distributed Renewable Energy Sources
DNO	Distribution Network Operator
DR	Data Rate
EV	Electric Vehicle
FIB	Flow Interference and Bandwidth Constraint
HV	High Voltage
ILP	Integer Linear Programming
ISP	Internet Service Provider
IoT	Internet of Thing
LP	Linera Programming
MAC	Medium Access Control
MEC	Mobile Edge-Cloud
MV	Medium Voltage
NFV	Network Function Virtualization
NLCG	Network Latency Constraint Greedy
OPEX	Operating Expenditure
OS	Operating System
PDC	Phasor Data Concentrator
PLC	Power Line Communication

PLeC	Path Length Constraint
PMU	Phasor Measurement Unit
P-SS	Primary-Substations
RIN	Route Interference Number
RTSE	Real-Time State Estimation
SCPA	Set Cover Partition Approximation
SDN	Software-Defined Networking
SFC	Service Function Chaining
SP	Service Provider
S-SS	Secondary-Substations
TCAM	Ternary Content Aware Memory
VALB	Virtual Auto-Scaling and Load Balancing
VM	Virtual Machine
VNF	Virtual Network Function
WAN	Wide Area Network

Publications

- Yang, B., Xu, Z., Chai, W.K., Liang, W., Tuncer, D., Galis, A. and Pavlou, G., **Algorithms for Fault-Tolerant Placement of Stateful Virtualized Network Functions**, In Communications (ICC), IEEE International Conference on, May 2018.

- Yang, B., Chai, W.K., Xu, Z., Katsaros, K.V. and Pavlou, G., **Cost-Efficient NFV-Enabled Mobile Edge-Cloud for Low Latency Mobile Applications**, IEEE Transactions on Network and Service Management, DOI: 10.1109/TNSM.2018.2790081, January 2018.

- Yang, B., Chai, W.K., Pavlou, G. and Katsaros, K.V., **Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud**. In Cloud Networking (Cloudnet), IEEE International Conference on, October 2016.

- Yang, B., Katsaros, K.V., Chai, W.K. and Pavlou, G., **Cost-efficient low latency communication infrastructure for synchrophasor applications in smart grids**, IEEE Systems Journal, DOI: 10.1109/JSYST.2016.2556420, May 2016.

- Chai, W.K., Wang, N., Katsaros, K.V., Kamel, G., Pavlou, G., Melis, S., Hoefling, M., Vieira, B., Romano, P., Sarri, S., Tesfay, T.T., Yang, B et al, **An information-centric communication infrastructure for real-time state estimation of active distribution networks**, IEEE Transactions on Smart Grid, 6(4), February 2015.

- Katsaros, K.V., Yang, B., Chai, W.K. and Pavlou, G., **November. Low latency communication infrastructure for synchrophasor applications in distribution networks**, In Smart Grid Communications (SmartGridComm), IEEE International Conference on, November 2014.

Chapter 1

Introduction

1.1 Context and Motivation

With the advancement of communication and computing technologies, ultra-low latency applications such as augmented reality (AR) [1], autonomous car control [2] and remote surgery [3] are expected to take off in the upcoming 5G era [4]. These applications require a low latency network to support fast interactive communications between users and servers. In this sense, designing novel communication networks to support low latency applications in a cost-efficient way has become a key research challenge.

Despite strong motivation in commercializing low latency applications, only a limited number of low latency networks have been deployed due to long wide area network (WAN) latencies. According to [5], the average WAN latencies range from 50ms to 83ms (e.g., round-trip time), which makes Internet Service Providers (ISPs) unable to provide low latency services (e.g., 10ms). The long WAN latencies are intrinsically due to the current Internet communication paradigm, in which a client sends a service request across the WAN to access services located at end servers [6]. In such a scenario, a request needs to go through a number of network equipment (e.g., servers, middleboxes [7], routers) along the communication path, where different types of delay might be incurred. For instance, a request might suffer from transmission delay due to low bandwidth resources on an intermediate network link. Further, a request might experience long queueing delay at an intermediate middlebox (e.g., Firewall) [8] due to the large volume of inbound traffic and lack of computing resources (e.g., CPU, memory). Similarly, long processing delay might be encountered at end servers due to inefficient allocation and scheduling of computing resources.

Existing low latency networks adopt overprovisioning to achieve ultra low latency at the cost of significant capital expenditure (CAPEX) and operating expenditure (OPEX). For instance, dedicated networks for financial trading and military communications adopt a full optical fiber deployment to achieve single-purpose low latency communications [9]. However, such a solution cannot be applied to large-scale commercial networks such as Internet of things (IoTs) [10] and 5G cellular networks due to the large number of costly high-bandwidth network links. Similarly, deploying more network equipment (e.g., servers) can increase the overall computing resources [11], thereby enabling fast data processing. As such, latencies can be guaranteed given that computational congestions (e.g., queueing delays at network equipment such as servers) in the face of peak workloads are avoided. However, the large number of deployed servers not only result in substantial deployment cost but also consume considerable energy during their operation. According to [12], energy consumption represents a major part of ISPs' operating expenses (OPEX), and it is expected to further increase by 10-12% per year [13]. Therefore, in order to make low latency applications accessible to the general public, low latency communication networks need to be carefully designed so that both CAPEX and OPEX can be minimized.

The advent of network function virtualization (NFV) [8] enables novel opportunities to achieve low latency, while largely improving the CAPEX and OPEX of networks. The fundamental and original idea of NFV is to exploit high-volume commodity servers at different network locations (e.g., base station, aggregation point, access router, core router, etc.) to provide virtualized resources (e.g., bandwidth resources and computing resources) in the form of virtual machines (VMs) [7]. As such, network resources are decoupled from dedicated hardware, and any commodity server can instantiate any network functions (i.e., middleboxes) from its VMs. NFV has been further extended to use its VMs to support more general service processing such as video transcoding [14]. By doing so, all the network equipment (e.g., middleboxes, routers, and servers) on the entire end-to-end communication path can be now virtualized. One direct benefit is the long-term CAPEX reduction. For instance, rather than purchasing new expensive dedicated hardware for new network functions or services, new functions/services can be installed as an instance of plain software into existing high-volume commodity servers. This avoids purchasing new and expensive dedicated equipment, and simplifies the service deployment process. In addition, resource usage efficiency and energy consumption can be largely improved (i.e., OPEX reduction) [15] by NFV. This is achieved by dynamically reallocating or migrating instantiated functions/services (e.g., hosted in VMs) to fewer commodity servers (e.g., shutting down low-utilized servers) during non-peak traffic time. At the same time, network functions and services can be instantiated at network locations that best serve end users in terms of access latency. For example, VM instances of end services can be brought from datacenters (DCs) to access points (APs) to shorten communication latencies, if APs are equipped with commodity servers.

Despite the abovementioned novel opportunities enabled by NFV, there are still many engineering challenges raised by the design of resource management approaches for cost-efficient low latency communications. For instance, existing underlying communication infrastructures often lack of high-bandwidth network links (e.g., backhaul network [16], smart grid [17]), which is the major rationale behind failing low latency. To this end, the cost-efficient (CAPEX) deployment/upgrade of underlying communication infrastructures with costly high-bandwidth links (e.g., WiMAX [18], optical fiber) is required. Next, in order to continuously maintain cost efficiency and low latency in a NFV-enabled network, virtualized resources on end-to-end communication paths need to be dynamically optimized in the face of varying network conditions (e.g., due to user mobility, varying workload), so that the allocated resources are always being efficiently utilized. This involves dynamically finding 1) the optimal placement of VM instances for intermediate network functions and end services, 2) the optimal VM capacity and 3) the optimal end-to-end routing paths that go through the instantiated VMs. In particular, due to the capacity limitations of commodity servers, the VM placement, capacity and routing paths need to be jointly determined to optimally utilize different network resources to achieve the required latency. On the other hand, the optimal resource allocation approach needs to ensure that low latency services can be continuously delivered even under extreme network conditions such as network failures. That is, when certain network equipment or VMs are unavailable due to faulty hardware or software, the back-up resources need to be in place to guarantee low latency.

1.2 Problem Statement

Given the abovementioned design challenges in cost-efficient resource management, the following questions will be addressed in this thesis.

1. How to plan network capacities (e.g., link capacities) for underlying communication infrastructures so that the deployment cost (CAPEX) can be minimized and low latency can be achieved.

2. How to design online end-to-end resource management algorithms, so that different end-to-end virtualized resources can be dynamically and jointly managed in the face of varying network traffic, while always achieving low latencies and maintaining low operational costs (OPEX).

3. How to further guarantee low latencies and low operational costs in face of network failures (e.g., hardware failures from underlying communication infrastructure, software failures in VMs).

These open questions motivate the design of a resource management framework that can effectively address the deployment costs (CAPEX) and operational costs (OPEX) for low latency communications. Specifically, the framework design questions will be answered through the analysis of realistic and specific low latency applications.

1.3 Contributions

Designing a resource management framework consists in devising optimization algorithms that optimally derive the best trade-off between the amount of allocated resource and the resulting end-to-end latencies. Specifically, algorithms that target different resource management scenarios such as static network planning, online resource allocation, and fault-tolerant resource allocation are studied. The details of contributions are presented in the following.

Design of cost-efficient low latency communication infrastructures [19, 20]: Given the high deployment costs of high-bandwidth technologies (e.g., optical fiber), the deployment/upgrade of low latency communication infrastructures needs to be optimized so that the required number of high-bandwidth network links is minimized. To this end, three different static network planning algorithms for cost-efficient low latency communication infrastructures were developed, aimed at minimizing deployment costs at the network planning stage. These algorithms determine the minimum amount of end-to-end network resources to achieve the required low latency, and derive network locations and capacities to deploy high-bandwidth communication links. Specifically, the proposed algorithms consider the characteristics of low latency applications (e.g., datarate, packet size) together with topological characteristics (e.g, path length and betweenness [21]) to identify network locations where network capacities are insufficient. In particular, one algorithm based on network calculus [22] provides worst-case guarantees on end-to-end latencies for deterministic workloads. Based on the proposed algorithms, a realistic case of upgrading smart grid communication networks [17] to support mission-critical applications is studied. The solution achieves 80% of deployment cost reduction compared to conventional approaches for a large set of real power grid topologies.

Dynamic cost-efficient resource management for low latency communications [23, 24]: The underlying communication infrastructures can be fully virtualized with the latest network virtualization technology [7], whereby operational costs

(OPEX) can be dynamically tuned by dynamic resource allocation (e.g., shutdown VMs during low workloads). In order to fully make use of the virtualized infrastructure to minimize operational costs, VM placement (e.g., VMs locations hosting end services), VM capacity and routing (e.g., network paths between users and services) need to be dynamically determined in face of varying network traffic. Conventional approaches [15, 25, 26] focus either on optimizing network link resources (e.g., bandwidth) or optimizing network node resources (e.g., CPU), and they assumed a set of predefined network locations to host VMs. In contrast, the dynamic resource management approach proposed in this thesis jointly optimizes different end-to-end resources (e.g., bandwidth and computing resources) to further improve the cost efficiency. Specifically, the proposed approach first applies a fast heuristic-based incremental allocation mechanism that dynamically increases the allocated resources in the network area where user traffic is heavy. Later, a reoptimization algorithm periodically adjusts the allocated resources to maintain a near-optimal operational cost over time. Mathematical analysis shows that the reoptimization algorithm provides a worst-case operational cost guarantee in polynomial time. Further, experiments under realistic network settings demonstrate that the dynamic resource management succeeds in achieving cost-efficient low latency communications. In particular, the dynamic approach continuously saves up to 33% cost efficiency compared to current approaches, while guaranteeing the cost efficiency to be within 20% of the lower bound of the optimal solution, regardless of network sizes, services' latency requirements and server capacities.

Fault-tolerant cost-efficient resource management [27]: The proposed design of underlying communication infrastructures and online resource management algorithms achieve a near-optimal cost efficiency while satisfying low latency requirements. However, both designs are vulnerable to hardware and software failures, which can lead to unavailable resources and application latency violations. To enhance online resource management's fault tolerance, a stateful fault-tolerant resource management problem is considered, whereby user states associated with VMs need to be transferred to the corresponding back-up VMs upon failures. In this problem, cost-efficient routing, VM placement, back-up VM placement and state transfer paths need to be jointly optimized. To this end, an efficient heuristic algorithm and a bicriteria approximation algorithm with performance (e.g, cost) guarantees are proposed. Specifically, the approximation algorithm adopts an auxiliary graph approach [28] to jointly consider all the on-path resources in an end-to-end manner. Simulations with large-scale networks show that the proposed algorithms largely outperform the conventional approaches where resources of network nodes and links are separately considered. Last, it must be stressed that the proposed solution is a general approach which is also valid for stateless fault-tolerant resource management.

1.4 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 provides the state-of-the-art in the area of cost-efficient resource management and low latency communications. Chapter 3 introduces the design of cost-efficient low latency communication infrastructures with a focus on reducing network deployment costs. Chapter 4 presents a dynamic resource management framework for low latency applications, aimed at achieving a trade-off between systems' operational costs and low latencies. Chapter 5 focuses on the fault-tolerant aspects of dynamic cost-efficient resource management. The proposed approach enhances networks' fault tolerance by simultaneously achieving cost efficiency and low latency.

Chapter 2

Related Work and Background

2.1 Introduction

Low latency communications have been extensively studied in dedicated single-purpose networks during the last decade. Conventional approaches to resolve this problem involve adopting high bandwidth technologies such as optical fibers to reduce latencies. However, these approaches raise concerns of cost efficiency and are not applicable to large-scale commercial networks due to considerable deployment costs. Alternatively, effective routing, resource allocation and scheduling approaches can prioritize packets with low latency requirements, so that the overall ratio of successful admitted packets that meet the latency requirements can be improved. This chapter looks into existing approaches in supporting low latency communications and cost-efficient resource management.

The remainder of this chapter is organized as follows. In Sec. 2.2, the background on low latency applications is presented. Specifically, different low latency applications' characteristics are investigated, followed by the introduction of end-to-end delay decomposition. Also, the latest technology enablers for low latency applications are discussed. In Sec. 2.3, the classic resource management models and optimization methods are presented. In Sec. 2.4, different types of network costs and resources considered in this thesis are presented. Then, the correlation between cost efficiency, low latency and resource allocation is discussed to further shed light on the problem and design space of cost-efficient resource management. Further, the related work of cost-efficient resource management is broken down into different subsections according to each subproblem, such as the design of cost-efficient low latency communication infrastructures; the design of cost-efficient dynamic resource management; and the design of fault-tolerant resource management.

2.2 Background

2.2.1 Low Latency Applications

Low latency applications have received significant attention in the last decade. The application scenario ranges from time-critical sensor-based monitoring applications to the latest smartphone applications such as face recognition and AR. In the following, three applications (AR, on-demand gaming, and smart grid monitoring applications) are reviewed as the representative applications.

- **Mobile augmented reality**: exploits computer vision to display relevant information as an overlay above a live view captured by camera [1, 29] of smartphones. For instance, additional information such as street names, restaurant ratings and number of parking places in a building could be added onto camera views to enhance user experience. However, such user experience enhancement relies on instant responses either from smartphones or remote clouds [30]. In the case of exploiting computing resources from remote clouds, smartphones' uploading flows consume significant bandwidth, which may lead to potential network bottlenecks. Given that humans are sensitive to delays and user enhancement is a real-time functionality, the response time requirement for mobile AR is therefore strict, and is on the order of hundreds of milliseconds [1, 29, 30].

- **On-demand gaming**: also known as cloud gaming [31], refers to online video gaming where gamers do not execute games at user terminals, but exploit computing resources at server/cloud side to perform computing expensive tasks (e.g., video transcoding). Specifically, rather than exploiting computing resources from resource-constrained terminals, on-demand gaming performs the intensive part of gaming computation (e.g., game graphics generation) remotely in clouds with the processed output streamed back to end users. Such shift from conventional gaming terminals to clouds/servers frees users from dedicated gaming hardware, but it requires the support of interactive low latency communications between users and gaming instances hosted in networks. Previous studies [31] showed that players begin to notice a quality degradation when RRT is more than 80ms.

- **Smart grid monitoring applications**: are of great importance to the next generation power grid [17]. In such networks, system dynamics such as voltage variation need to be monitored with high-frequency sampling rate [32]. By doing so, the real-time global network status is known to the power grid controller, which can further perform prompt control actions to protect power grids. However, these monitoring and control actions have to rely on low latency communication networks. In particular, time-critical applications running on communication networks need to be delivered within very stringent latency constraints as information exchanged between grid components is useful/valid only within a predefined time window as small as 3ms. Any failure of meeting such delay requirement could result in cascading failures and large-scale blackout [33].

As can be observed from the abovementioned, each application has its own characteristics that require a specific approach to reduce end-to-end delays. For instance, for time-critical smart grid applications, network delays on network links are the components that need to be reduced due to high bandwidth consumptions. In contrast, for gaming and mobile AR, processing delays at end servers have to be considered along with network delays due to the fact that these applications require intensive CPU and GPU processing.

2.2.2 End-to-End Latency Decomposition

In order to shed light on potential methods to achieve low latency, we give the identification of the various components of end-to-end delay, denoted as T_{e2e}. Further, we illustrate how delay components are grouped together to form the end-to-end

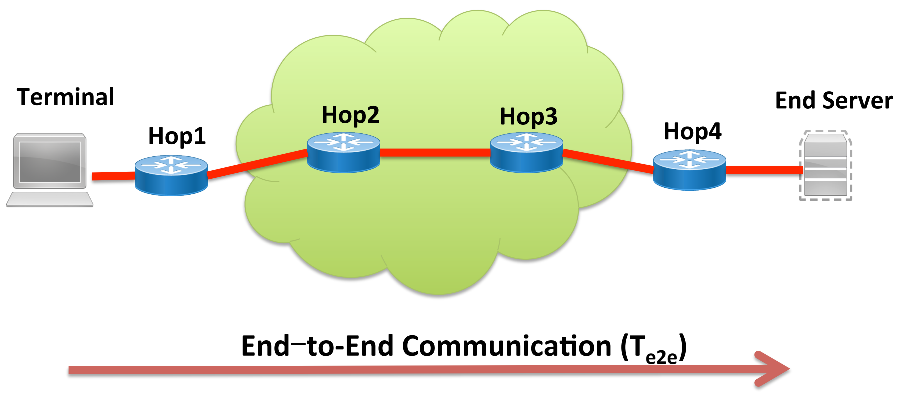


Figure 2.1: End-to-end communication example.

delay.

- Processing delay: denoted as t_{proc}, the time used for operations such as medium adaptation, (de)coding, switching, routing, message authentications codes generation / verification.
- Propagation delay: denoted as t_{prop}, depends on the transmission medium and the distance traveled by the signal.
- Transmission delay: denoted as t_{trans}, the time required to transmit the data and is subject to the bandwidth of the underlying transmission technology.
- Queuing delay: denoted as t_{queue}, the time spent by data waiting for transmission and processing at the transmitting devices. For instance, the computing congestion at devices is due to the lack of computing resources, and it is a consequence of queuing delay at devices.

To clarify how end-to-end delay is composed by delay components, a detailed example is given in Fig. 2.1. We see that an end-to-end communication is composed of per-hop communication at each of network node (e.g., hop1, hop2, etc), which can be further decomposed into the abovementioned delay components. For instance, an end-to-end communication path from the two communication terminals is shown by a red bold line across routers. A packet in such example travels from the sender machine to the next intermediate router, experiencing processing, propagation, transmission and queueing delays. Depending on the network protocols and the specific functionality, processing delay varies at each hop (i.e., processing delay in the terminals is higher than that of intermediate routers as application-level processing is involved). Furthermore, queueing delay is not necessarily experienced at each hop, and this depends on network conditions (e.g., network congestion), as well as workloads at the processing device.

2.2.3 Enabling Technologies

Virtualization technology [34] plays a key role in enabling cost-efficient low latency communications. By definition, virtualization refers to the act of creating a virtual version of computing resources (e.g., CPU, RAM, storage, etc), operating systems (OSs) and virtual networks. Such virtualization technology facilitates the management of physical resources through resource abstraction and virtual resource manager [7]. Also, the resulting monetary and operational costs can be dynamically optimized through resizable and migratable VMs. In the following, the related virtualization concept such as cloud computing, mobile edge computing, NFV and software-defined networking (SDN) is presented together with their applicability to cost-efficient low latency communications.

Cloud Computing: has become the predominant technology for hosting Internet applications and services in the last decade, where virtualized resources are grouped in the form of VMs and utilized to support computational tasks (e.g., transcoding) that used to run in physical machines. The major advantage of cloud technology is its intrinsic benefits brought by DC consolidation [30, 35], whereby the high resource utilization and concentration exploit economies of scale and lower the marginal cost of system administration and operations. In addition to consolidation, the other major advantage is the elastic control of virtual resources where the allocated virtualized resources can be elastically adjusted in the face of dynamic traffic. By doing so, cloud users can significantly reduce their CAPEX and OPEX as resources are consumed and charged only based on the actual consumption (i.e., no resource wastage).

Mobile Edge Computing [36]: Unlike cloud computing where virtualized resources are located at remote DCs, mobile edge computing exploits mobile edge-clouds (MECs) (e.g., micro-clouds) that are installed at network locations (e.g., APs, aggregation points) close to end users. As such, mobile users offload computationally expensive tasks to MECs' VMs where task processing takes place. By doing so, communication delays can be largely reduced compared to offloading tasks to remote DCs. However, in order to achieve low latencies, edge computing requires the deployment of a large number of micro-clouds, which breaks the DC consolidation and incurs significant operational costs [30]. To this end, cost-efficient resource management approaches for edge resources are required.

Network Function Virtualization: on the other hand, provides virtualized network functions (VNFs) that used to be embedded in dedicated network appliances such as firewall, load balancer and deep packet inspection (also referred as middleboxes) [8]. It aims to transform the way that network operators architect their networks by evolving standard IT virtualization technology to consolidate many network equipment types onto high volume commodity servers [8]. Such transformation towards VNFs enables efficient network resource sharing, whereas cloud computing and mobile edge computing enable computing resources sharing. Therefore, the joint use of NFV and cloud-related technologies is the key enabler for end-to-end cost-efficient resource management. In addition, when VNFs are interconnected, a service function chaining (SFC) [37] can be built which provides network processing as a chain of network functions. An important feature in SFC is the ordering of different VNFs, that is, network flows need to follow a specific order defined by ISPs or service providers (SPs) before they reach the end service.

Software-Defined Networking: was proposed to facilitate network configurations by decoupling data plane and control plane [38]. In SDN, network states such as network devices' utilization are continuously monitored by software-based controllers, which can further implement network intelligence for routing configuration decisions. Such configuration decisions are sent back via a programmable open interface [39] to virtualized network devices (e.g., virtual switch [40]) to perform optimized packet forwarding. In contrast, traditional network devices have a combined data plane and control plane, which makes the implementation of new network routing a difficult task. In the context of dynamic resource management, the role of SDN is to enable dynamic network routing, so that network resources connecting different virtual network components (e.g., clouds, MECs, virtual switch and VNFs) can be efficiently managed.

The abovementioned virtualization technologies provide virtualization at different network locations to enable intelligent end-to-end resource management approaches. For instance, cloud computing and mobile edge computing provide end service virtualization, whereas NFV provides virtualization for intermediate network equipment. On the other hand, SDN provides the routing control between each virtualized network entity.

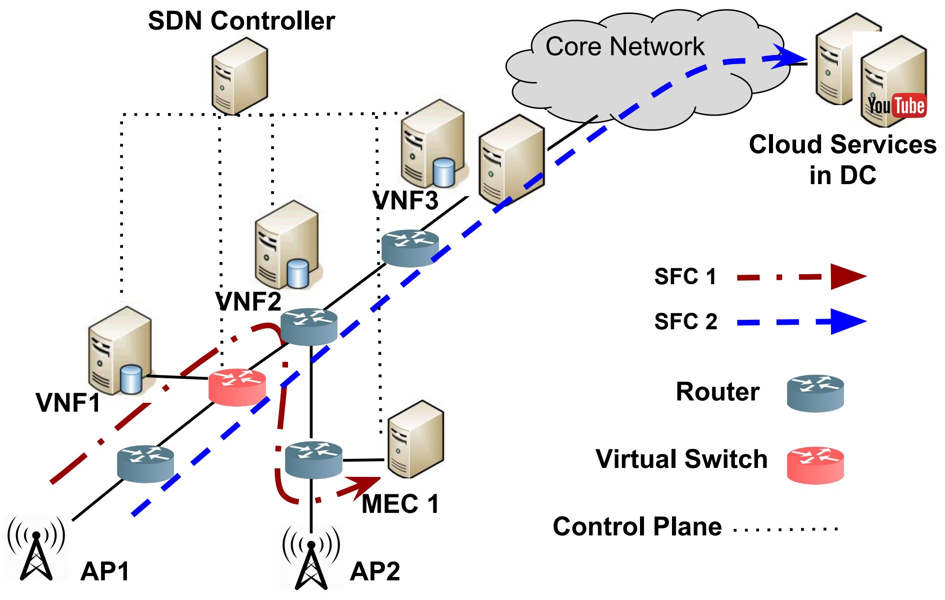


Figure 2.2: End-to-end routing and resource allocation.

Fig. 2.2 shows an example of end-to-end communications enabled by different virtualization technologies. Two end-to-end communication paths are presented, whereby Path 1 goes through SFC1 (e.g., composed of VNF1 and VNF2) to reach MEC1, and Path2 goes through SFC2 (e.g., composed of VNF1, VNF2, and VNF3) to reach the remote DC. In these examples, the computing resources and bandwidth resources can be managed (e.g., scale up or scale down) in an on-demand manner following the variation of traffic.

2.3 Resource Management in Computer Networks

Given the underlying communication infrastructure and the upper layer virtualization, different network resources can be managed in a dynamic manner. In particular, managing resources involves deriving the location and amount of network resources to be allocated. In this section, we firstly introduce different types of network resources and costs considered in this thesis. Then, the correlation between latency, resource and cost is discussed to shed light on potential resource management models and solutions.

2.3.1 Resource Types

- **Computing resources** refer to resources such as CPU, GPU and RAM. When a service is executed in a computer system, a number of processes are initialized, which occupy CPU cycles and RAM for computation. Furthermore, the utilized CPU and RAM consume energy (e.g., operational cost), which is highly correlated with resource utilization.

- **Storage resources** refer to resources such as disk space of computer systems, which are used to store data generated during service operation. For instance, certain services may record system logs and user-related data, which is stored in systems' disks.

- **Bandwidth resources** are allocated through network interface cards, which determine the communication throughput.

2.3.2 Cost Definition

In the following, the definitions of costs considered in this thesis are presented.

- **Deployment costs**: referred to as CAPEX, represent costs related to the expenses on equipment such as network cables, routers, APs, gateway, middleboxes and servers, as well as expenses incurred in equipment deployment [41]. Specifically, the overall deployment costs highly depend on the

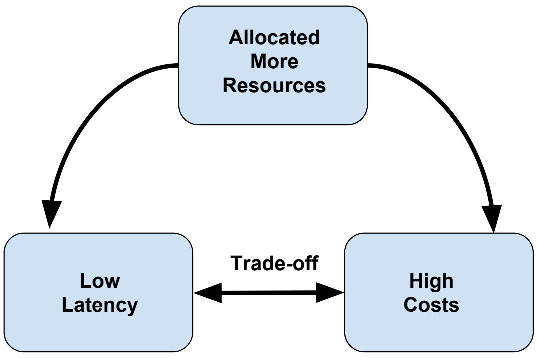


Figure 2.3: Trade-off between latency and cost. Allocate more resources reduces latency, but results in high costs.

network scale, which can be interpreted as the number of aforementioned network equipment. As such, the deployment plans (e.g., static network planning) highly affect the resulting deployment costs, and a careful design of communication infrastructures that minimizes the number of network equipment can effectively reduce deployment costs.

- **Operational costs**: refer to costs related to expenses incurred during network operation, which include staffing costs, energy consumption at network equipment (e.g., servers, APs, routers, etc) and network management costs. For instance, according to [42], a DC consumes as much energy as 25,000 households, representing the most significant part of ISPs' OPEX. On the other hand, the impact of network management overheads (i.e., the number of management messages, the time required to configure a network, bandwidth consumption) also leads to operational costs. However, these costs can only be implicitly interpreted as monetary impact to network operators' revenues. For example, considerable management overheads could result in network congestions, which in turn affect the network performance and operators' revenues. To solve the cost issues, dynamic approaches such as temporarily shutting down low utilized physical machines or changing routing paths could largely reduce operational costs.

2.3.3 Correlation Between Latency, Resources, and Costs

Having specified different types of costs and delay components, the correlation between these two factors is introduced from a resource allocation's perspective. First, the correlation between costs and provisioned resources is straightforward, that is, the more resources (both computing and bandwidth resources) are allocated to a service, the more cost it incurs (see Fig. 2.3). For example, data-intensive low latency applications need considerable computing resources to achieve low processing delay, but considerable computing resources usually result in high operational costs due to high energy consumption.

The correlation between provisioned resources and resulting latency is less obvious, but it follows the same principle as the correlation between costs and resource provisioning. Specifically, the more resources are provisioned, the less latency a network packet experiences (see Fig. 2.3). For instance, processing any packet requires computing resources, and the speed of processing a packet is proportional to the allocated resources. In other words, the more CPUs are allocated, the faster a task will be computed. Similarly, exploiting high capacity bandwidth technology such as optical fiber, as opposed to low bandwidth technology, enables low transmission delay and low propagation delay.

Given the aforementioned correlation between latency, provisioned resources, and costs, it is now clear that provisioning more resources on the communication path can improve the resulting end-to-end latency, but leads to concerns in terms of costs. Given such trade-off, the problem of supporting cost-efficient low latency communication can be transferred into a cost-efficient resource management problem; that is, how to efficiently allocate network resources (e.g., computational resources, network resources) in a cost-efficient manner to guarantee low latency requirements.

2.3.4 Relevant Optimization Models

Optimization models provide the fundamental formulations for conventional resource management problems, based on which advanced resource management problems (e.g., with more constraints) can be formulated. Originally, these models are designed to solve operational research problems, whereby the locations and capacities of warehouses need to be determined so that commodities of different sizes can be transported to warehouses with minimum costs [43]. Later, optimization models have been largely adopted in the design of telecommunication networks and computer networks [44]. For instance, the locations of end servers and the selection of routing paths can be represented by a set of integer decision variables. The potential optimization algorithm determines a set of server locations and routing paths based on problem inputs and objective. A typical problem input can be in the form of a set of user requests, which need to be routed and processed in the considered network. A typical objective can be the minimization of maximum bandwidth utilization, that is, minimizing the level of network congestion.

- **Facility location**: is one of the most common models in both static and dynamic network resource allocation [43]. It considers a set of potential locations for warehouses with fixed costs and capacities, and a set of customers with demands for goods supplied from these warehouses. The transportation cost per unit for goods supplied from warehouses to all customers is given. The problem is to derive the locations of a subset of warehouses that minimize the total costs so that all customers can be satisfied without violating the capacity constraints of warehouses. At the same time, the problem needs to find the assignment of customers to facilities, which can be referred as transportation problems [45]. In order to adapt the facility location model to computer networks, a few changes need to be made. First, warehouses need to be replaced by servers. Then, customers need to be replaced by users, and transportation costs need to be represented by communication costs between servers and users.

- **Set covering location**: is the extension of the well-know set covering model [46]. In the set cover location, it finds the minimum number of facility locations that cover all customers within a distance constraint. A variant of the set covering problem, capacitated set covering problem (CSCP) [47], that takes the set capacities into account, can be applied to many network planning scenarios. For instance, the minimum number of required servers to support a certain number of users can be formulated with CSCP, whereby each server has a capacity constraint, and each user has certain request demands that need to be served within a latency constraint.

- **Multi-commodity flow**: considers a routing problem whereby a certain number of commodities need to be transported from a set of source nodes to a set of destinations without violating all link capacities [48]. Specifically, the routing decision variables can be either integer or linear variables, which correspond to non-splittable routing or splittable routing in a network. The multi-commodity flow model provides the basis for more advanced routing models where additional constraints (e.g., latency constraint, single source) can be expanded. For instance, [49] extended this model with an additional condition, whereby flows need to be processed by intermediate nodes in the network. As such, the placement and resource allocation of in-network processing (e.g., middleboxes, SFC) can be taken into account.

2.3.5 Optimization Solutions

Once resource management problems are formulated, a decision space consisting of a set of potential decisions will be created. For instance, an optimal routing path might need to be derived between two nodes in a network in order to achieve the lowest communication latency. To solve this, all possible paths between the two nodes will be first found to create a solution space, from which the optimal path will be searched. However, the optimization solution space in computer network resource management problems could become extremely large given the large problem input (e.g., a city can have millions of users, and a very complex computer network topology). As such, exhaustive search (e.g., brute-force [50]) is not often an effective and feasible solution to the formulated problems due to its complexity. To this end, efficient optimization strategies such as relaxation, meta heuristic and approximation can be applied to either reduce the problem complexity or accelerate the optimization searching process. The major difference between each intelligent

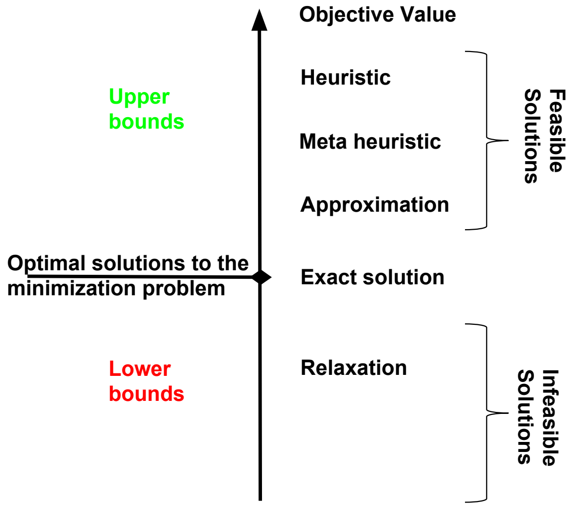


Figure 2.4: Illustration of upper and lower bounds for minimization problems.

approach lies in the achieved optimality and the optimization running time. That is, how far the derived solution is from the optimum and how quick the solution can be obtained. In the following, two important concepts, lower bound and upper bound [50], are first introduced, which can be used to classify the abovementioned optimization solutions (see Fig. 2.4).

- **Lower bound**: refers to the solution space that achieves a smaller overall objective value (when the optimization problem is a minimization problem) compared to the optimum (see Fig. 2.4). However, such solutions are derived by omitting a certain constraint (e.g., relax integer constraints to linear constraints). As a result, the obtained solutions are not feasible solutions to the original problem. Typical approaches in constraint relaxation include linear programming (LP) relaxation and Lagrangian relaxation [51], which will later be discussed in more details.

- **Upper bound**: refers to the solution space that achieves a higher overall objective value (when the optimization problem is a minimization problem) compared to the optimum (See Fig. 2.4). Unlike lower bound where the solutions are not feasible, upper bound solutions are found by searching through the feasible solution space that satisfies all constraints of the original problem.

Fig. 2.4 provides an overview of the mapping between each optimization solution and the achieved optimization performance. Clearly, most of the existing solutions (e.g., heuristic, meta heuristic and approximation) look for a solution in the feasible solution space. In contrast, relaxation-related solutions achieve a better solution than the optimum, but cannot justify the feasibility of the obtained solutions. In the following, each optimization solution is discussed with its advantages and disadvantages.

- **Relaxation**: refers to methods that solve a simplified version of the original problem (e.g., relax a certain constraints). By doing so, a complex combinatorial optimization problem can be quickly solved. However, this obtained solution is not a feasible solution to the original problem. In order to obtain feasible solutions, the relaxed solutions need to be adjusted (e.g., in the case of integer linear programming (ILP) relaxation, the obtained linear solutions need to be rounded up or down) [50]. In addition, the relaxation with rounding techniques is widely used in deriving exact [52] and approximate solutions [53].

- **Exact solution**: solves the formulated problem in an optimal way. It either adopts an existing solver (e.g., CPLEX [54]) or approaches from brute-force enumeration (e.g., branch-and-bound, branch-and-cut, branch-and-price) to derive the set of optimal decisions [50]. However, exact solutions can be only applied to small problem instances as most of network resource management problems are ILP, and are therefore NP-hard [50].

- **Meta heuristic**: Given the problem input size and the resulting combinatorial solution space, the searching space might be extremely large, which motivates intelligent ways of searching optimal solutions in the problem space. In this sense, Meta heuristic solutions are devised to find near-optimal solutions by always keeping improving a candidate solution with regard to a given measure of quality (e.g., objective value). Existing meta heuristic includes simulated annealing, genetic algorithms, ant colony optimization, tabu search and etc [55].

- **Heuristic**: is a technique to quickly solve a complex optimization problem. The low execution time is achieved by sacrificing the optimality of solutions. Heuristic is a common approach to solve in resource management problems due to their complexity.

- **Approximation algorithm**: refers to algorithms that provide a performance guarantee in polynomial time. Specifically, it provides a resulting performance that is at most constant times of the optimum. However, it is challenging to prove that an approximation algorithm has a performance guarantee.

In this thesis, the exact solutions, heuristics and approximation algorithms are thoroughly investigated for the design of resource allocation algorithms. In particular, we aim to provide performance guarantees for ISPs with approximation algorithms, so that the worst-case network costs can be taken into account for ISPs' networks.

2.4 Problem and Design Space

Having specified different optimization models and solutions, we now investigate the design challenges of a cost-efficient resource management framework. Essentially, this consists in finding the optimal allocation of different resources (e.g., computing resources, storage resources, and bandwidth resources) to achieve a certain required latency.

Fig. 2.5 shows an example of different subproblems considered in this thesis. For instance, the lower part of Fig. 2.5 illustrates the design problem of cost-efficient underlying low latency networks, whereby the network capacities and the locations to install/deploy different network equipment (e.g., server, MEC, DC, etc) need to be optimally derived. Furthermore, the upper part of Fig. 2.5 provides an intuitive example of the dynamic end-to-end resource management and the fault-tolerant network resource management for low latency communications. For the dynamic end-to-end resource management problem, network resources (bandwidth and computing resources) on two end-to-end communication paths need to to dynamically allocated (see Fig. 2.5 the red path to MEC and the blue path to DC). Problems of this category not only consist in deriving the locations to instantiate virtualized resources, but also in finding the amount of required resources to guarantee low latency requirements (e.g., the portion of shared computing/bandwidth resources in Fig. 2.5). Moreover, the back-up resources need to be allocated to prevent from failing low latencies during network failures (e.g., network links and node in green, see Fig. 2.5). In particular, the resulting costs of back-up resources need to be optimized.

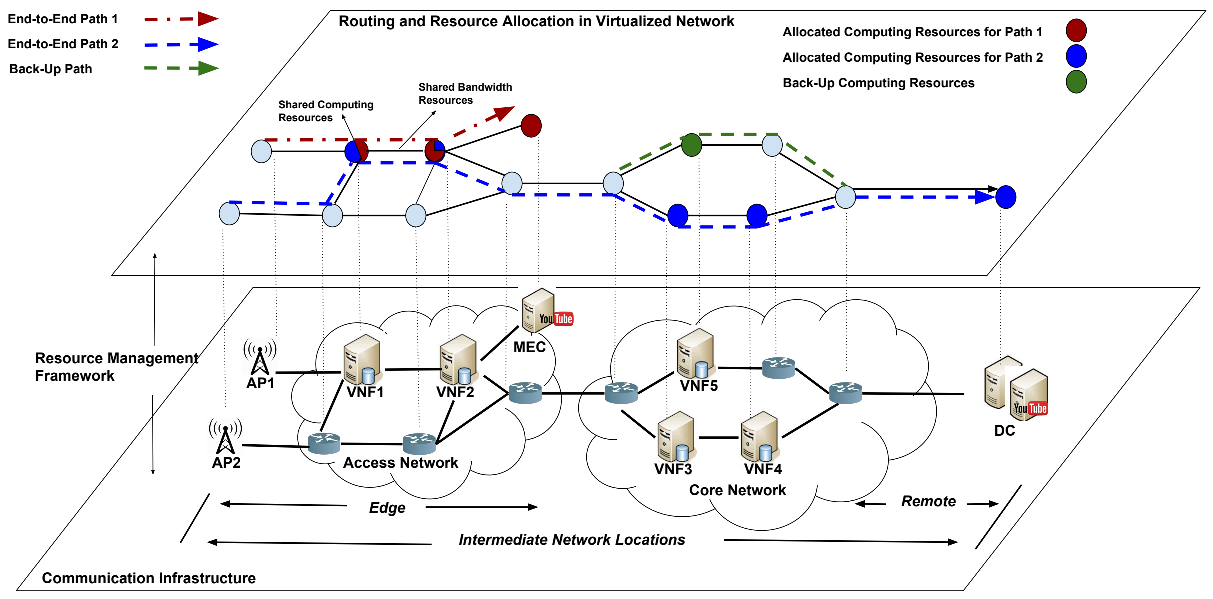

End-to-End Path 1
End-to-End Path 2
Back-Up Path
Routing and Resource Allocation in Virtualized Network
Allocated Computing Resources for Path 1
Allocated Computing Resources for Path 2
Back-Up Computing Resources
Shared Computing Resources
Shared Bandwidth Resources
Resource Management Framework
AP1
AP2
VNF1
VNF2
MEC
VNF5
VNF3
VNF4
DC
Access Network
Core Network
Edge
Remote
Intermediate Network Locations
Communication Infrastructure

Figure 2.5: End-to-end resource management framework.

2.4.1 Static Network Planning

The first step towards a cost-efficient resource management framework is to provide low latency communication infrastructures for ISPs. This consists of finding the optimal placement of physical network equipment such as router, server, network link and etc (see the lower part of Fig. 2.5). Conventional approaches to solveing the low latency network planning problem (i.e., guarantee worst-case latency) adopt overprovisioning [56], whereby the capacity of network equipment is provisioned according to the predicted peak workload. However, ISPs' networks are not fully utilized during off-peak times as the average level of workload is much smaller than the peak workload [57]. Obviously, overprovisioning approaches result in resource wastage, and are not cost-efficient. To solve this, a trade-off between cost efficiency and low latency needs to be made in the design of the underlying communication infrastructures.

To achieve the abovementioned trade-off, advanced techniques focused on operational research [58] and graph theory [59], have been applied at different stages of network planning. First, when networks need to be designed from scratch, facility location, set cover location and transportation models are largely adopted to model the decision-making problem with respect to the locations of network links and servers [44]. In addition, capacitated models such as capacitated facility location [43] and capacitated set cover location [47] are adopted to formulate not only the locations but also the capacities of network links and servers. Since network planning takes place at the design stage, there is no actual requirement in terms of optimization algorithms' running time. That is, network planning optimization is offline optimization, and can, therefore, afford long running time incurred by exact solutions. Second, when existing networks need to increase their capacity to accommodate higher traffic [60], decisions such as where to deploy additional network links to increase bandwidth resources need to be made in an efficient way, such that the network upgrade costs (e.g., required additional network equipment) can be minimized. To solve the decision-making problem, a network performance analysis with respect to network congestion locations is required to first understand the demanded capacity of the considered network. Further, an analysis of the expected network traffic after network upgrade needs to be performed in order to provide the capacities.

An extensive amount of work has been carried out to solve network planning problems (e.g., design from scratch) [61, 62, 63, 64]. They adopted exact solutions (e.g., brute-force approach) with CPLEX optimizer to derive the locations and capacities of network equipment respectively in the context of smart grid, wireless sensor network, DC network and mobile edge network. However, the exact solutions can only solve small-scale complex planning problems or simple network planning problems in reality. As such, the large-scale complex network planning problem (e.g., IoT) will lead to infinite optimization running time if exact solutions are adopted. Compared to network planning problems from scratch, the network upgrade problem has received little attention, and is considered to be more complex due to the additional constraints imposed by existing network topologies. In this sense, analysis with advanced graph theory and complex network theory [65] will be required to first understand the problems faced by existing networks.

In Sec. 3, we proposed a novel network upgrade approach specifically targeting cost-efficient upgrade problems for low latency communication networks.

2.4.2 Dynamic End-to-End Resource Management

Given the underlying communication infrastructures and the latest advancements in virtualization, different network resources (e.g., bandwidth resources, computing resources and storage resources) on end-to-end paths (see the upper layer of Fig. 2.5) can be jointly optimized to achieve cost-efficient low latency networks. Specifically, such optimization process involves simultaneously determining resource allocation on end servers and routing paths between source network nodes and destination network nodes. In the following, resource allocation problems are classified based on locations where allocation takes place. First, resource allocation at remote DC (e.g., cloud servers) is reviewed (see Fig. 2.5). Second, resource allocation in the context of mobile edge computing will be reviewed whereby MECs are located at network edges (see Fig. 2.5). Last, routing algorithms that derive the paths between users and end services are reviewed (see Fig. 2.5).

Cost-efficient Resource Allocation in Cloud Computing: Cost efficiency [15, 66, 67, 68, 69, 70, 71, 72, 73, 74] has been extensively studied in the context of cloud computing over the last years. Most of the work [15, 69, 71, 72] in this domain focused on achieving cloud consolidation by dynamically allocating/reallocating VMs that host end services, thereby minimizing the number of active servers. The other direction in improving cloud cost efficiency considers monetary costs for cloud users [66, 68, 70, 73, 75]. In the first category, [15, 69, 71, 72, 74] studied the energy consumption minimization problem in a single DC. The objective of these problems can be either the minimization of a number of active servers [69, 71, 74] or the minimization of the resulting power consumption [15, 72]. Unfortunately, work in DC consolidation focused on the reduction of energy consumption, and did not consider computational tasks' deadline (this will be discussed later), which is a key requirements for low latency services.

In contrast to the abovementioned studies, work in deadline-constrained auto-scaling and scheduling [56, 68, 73] focused on achieving latency requirements of cloud services via elastic resource allocation and scheduling. This class of work considered task completion deadlines. They aimed to derive the minimum allocated computing resources and the optimal sequence of task processing (e.g., prioritize packet processing for packets with lower latency) to efficiently meet task completion deadlines. However, since network delays dominate the entire end-to-end delay, the achieved latency savings in DCs with scheduling techniques are limited compared to latency savings achieved by MECs, which largely reduces network delays.

Resource Allocation in Edge/Fog Computing: Unlike providing services from remote DCs, edge/fog computing aims to bring services closer to end users by exploiting virtualized resources in micro-clouds located at network edges. By doing so, the communication latency can be largely reduced, but it raises concerns in cost efficiency due to the distributed nature of micro-clouds and the break of DC consolidation. Limiting the number of distributed clouds can resolve the cost issue, but would result in the violation of low latency requirements. In this sense, a trade-off between the amount of allocated resources and delays needs to be addressed (i.e., the more resources are allocated to a service, the faster the processing will be).

Most of the recent work in MEC resource allocation [25, 76, 20, 77, 78] considered that end services have already been placed/instantiated in VMs from MECs, and the uploading of application logics is not required. As such, mobile users can simply upload traffic to MECs to be processed. These studies investigated the service placement, network planning, dynamic resource allocation and user admission problems. Compared to similar problems in the context of DC-based cloud computing, the two distinguishing features, resource limitation and strict latency constraint of edge computing systems, need to be carefully considered. Existing work such as [78] considered distributed service placement for multiple services in resource constrained MECs. The authors adopted a mixed integer programming to formulate the problem, and aimed to find the service placement that minimizes admission failures. Then, they solved it with heuristics inspired by caching content placement heuristics [79]. [80] studied a MEC planning problem where they formulated a slightly different facility location problem that has a predetermined K MECs to be placed. In this problem, both locations of MEC and routing paths need to be determined such that the average communication delay is minimized. [81] considered an admission control problem in computing resource-constrained MECs. In particular, they adopted the distribution of arriving mobile users and the average MEC service rate to model the gained utility of admitting a mobile user with a Semi-Markov decision process. Next, they integrated the Markov decision model into a LP problem formulation, and solved the problem with existing LP solver. [82] considered load balancing in MEC whereby they devised two dynamic workload-to-MEC assignment algorithms (heuristic and genetic algorithm) to minimize the maximum average response time in all MECs.

Most of the existing work in edge computing focused on minimizing end-to-end latencies with fixed-location micro-clouds. However, the trade-off between achieved latency and the cost efficiency of edge cloud resources has not been studied. At the same time, the potentials of jointly exploiting dynamic routing and dynamic resource allocation on end-to-end communication paths have not been revealed. To this end, we review existing routing approaches in the following.

Dynamic Request Routing : Routing aims to find network paths between two end pairs that minimize the accumulated metrics of traversed paths. Depending on the optimization objective, such metric can be defined in different forms such as congestion-caused monetary loss, link latency or amount of available link bandwidth. Conventional routing algorithms such as distance vector algorithms and link-state algorithms adopt different searching methods to find shortest paths between a set of paired locations [26]. However, in order to apply routing policies at routers, network operators need to separately configure each on-path router with low-level and often vendor-specific commands [7], which is difficult to achieve in the current Internet paradigm (e.g., equipment from different vendors).

SDN decouples the data plane from the control plane to facilitate the implementation of complex routing decisions. As such, SDN can more intelligently and dynamically optimize the use of network resources compared to conventional approaches. Most of the work to this regard focused on optimal routing approaches that optimize certain objective functions such as (e.g., minMax link utilization, maximum request admission rate, minMax server utilization). For instance, [83, 84] considered a routing optimization problem in SDN, whereby they aimed to find routing paths that maximize the admitted flows while conforming to SDN forwarding table size. Specifically, the problem of Ternary Content Aware Memory (TCAM) was taken into account, which is a limited and expensive resource. As a result, the number of flows that can go through a TCAM-based router is constrained by the forwarding table size. To solve the problem, [83] adopted a randomized rounding approach that provides a performance guarantee on the overall admitted flow. [84] adopted a graph theory approach to construct an auxiliary graph converting network node capacity constraints (e.g., forwarding table size constraint) to link constraints (e.g., using the capacity of an added virtual link to represent the node capacity).

Summary of dynamic end-to-end resource management : Clearly, resource allocation in DCs does not entirely resolve latency issues for low latency applications (i.e., extreme low latency requirements cannot be met). In contrast, edge computing addresses latency issues, but faces issues in cost efficiency. This requires a joint optimization of dynamic routing and dynamic resource allocation at MECs. Existing work in routing with TCAM constraints firstly introduced the concept of jointly optimizing resources on network nodes and links. However, the joint optimization of resources from end cloud servers and network links has not yet been addressed. As such, it might occur that abundant amount of on-path network bandwidth is allocated to an end-to-end communication path, but an end cloud server does not possess enough computing resources to process the routed traffic [85]. As a result, the provisioned on-path resources are not efficiently utilized (i.e., resource wastage) due to the bottleneck at the end server, which would lead to latency violations.

In Sec. 4, we solved the online cost-efficient resource management problem for ISPs by adopting MECs. Specifically, we aim to dynamically minimize the resulting operational cost of different network resources while always achieving the required low latencies by optimally exploiting different resources.

2.4.3 Fault-Tolerant End-to-End Resource Management

The faulty hardware and software could severely affect communication latencies. Therefore, we consider a specific scenario in cost-efficient end-to-end resource management, which is the resource allocation in SFC. In this scenario, different VNFs such as deep packet inspection, firewall, and load balancer are chained together to provide more complex network services. As such, for services that need to go through SFCs, the end-to-end latency will highly depend on the placement of VNFs, requests-to-VNFs assignment and the routing between VNFs.

[86] considered an ordered VNF placement and routing problem in a network resource-constrained environment. In this problem, the authors considered three objectives when optimizing the VNF placement, aimed at balancing the resulting load on network links, minimizing the number of used network nodes for hosting VNF instances and minimizing end-to-end latencies of the created paths. [87] considered an unordered VNF placement and routing problem for operational cost minimization, whereby they adopted a facility location model and a general assignment model to respectively formulate the VNF placement and VNF request assignment. To solve it, they proposed an approximation-based algorithm that leveraged linear programming relaxation and rounding techniques to find near-optimal solutions. [88] aimed to find both the optimal VNF placement and the assignment of requests to VNF chains. They first considered a maximum network link utilization minimization problem, and then considered an energy minimization problem. A

heuristic algorithm was proposed to address the formulated problem. Clearly, none of the existing work jointly considered the correlation between the load on network nodes (e.g., VNF utilization) and network links. Such joint consideration is essential in the upcoming 5G low latency end-to-end communications as any node or link congestion could potentially affect the perceived user experience.

The abovementioned studies assumed that VNFs are stable, where both software and hardware can operate without failures. However, in practice, VNF failures might frequently occur, due to the variety of reasons, such as connectivity errors (e.g., link flaps, device unreachability, port errors), hardware faults (memory errors, defective chassis), misconfiguration (wrong rule insertion, configuration conflicts), software faults (reboot, OS errors) or excessive resource utilization [89]. If such failures are not handled seamlessly and correctly, they will introduce great degradation of service performance and reliability. Most studies on providing fault tolerance support for NFV-enabled networks have been focusing on either designing and implementing systems with fault tolerance support [90, 91, 92] or plan-stage VNF placements based on statistical methods [93]. [93] investigated the problem of NFV backup instances deployment problem, given failure probabilities of different network functions. Most of these studies, however, are not clear how to jointly route user requests and place active and stand-by instances of their service chains, such that a specific network performance is optimized.

Unlike these studies, in Sec. 5, the joint routing and placement of their active and stand-by instances is studied, such that end-to-end resources are optimized, while the lowest operational costs are achieved and latency constraints are satisfied.

Chapter 3

Cost-efficient Network Planning for Low Latency Applications

In this chapter, we first look at issues in designing cost-efficient low latency infrastructures, which consist of the offline networking planning/upgrade of communication links. We formulate this physical communication infrastructure design problem with ILP as a placement problem, and propose three graph theory based algorithms aiming to achieve both the cost efficiency (e.g., deployment costs) and the required low latency. In particular, this chapter investigates a case study in smart grid communications, whereby delay-sensitive applications are vital to the reliability of power grid systems. The main contributions consist of three network planning algorithms that adopt topological characteristics to optimize network operators' deployment costs.

3.1 Introduction

The energy sector has been undergoing major transformative changes in recent years in order to address pressing concerns in improving energy efficiency of the grid and to reduce overall carbon emissions. The increasing penetration of distributed renewable energy sources (e.g., solar/wind farms), the rising deployment of electric vehicles [94, 95] and active consumer participation into power grid operations (e.g., interactive consumer applications) are pushing today's power grid infrastructure to the limit. The progressive integration of these active components introduces significantly higher system volatility, posing new challenges to system stability, with respect to power quality, voltage regulation, protection [96] and fault location. In fact, this constitutes a major shift from passive to active distribution networks (ADNs)[1].

To cope with this increasing volatility, distribution network operators (DNOs) aim at the design and development of enhanced cyber-physical systems enabling both the fine-grained *monitoring* and *control* of their power grid infrastructure. In the envisioned systems, a communication infrastructure supports the near-real time *observability* of the power grid conditions, enabling in turn the control of the power grid infrastructure in terms of the aforementioned control operations. In this context, the deployment of high-precision Phasor Measurement Units (PMUs) [97] gains a significant role for DNOs. By supporting high rate, synchronized monitoring of key system parameters, PMUs enable the synchrophasor-based real-time state estimation (RTSE) [98] of the grid, opening the way for fine-grained and timely control of the overall system [99]. For example, fault localization enables the instant identification and the subsequent opening/closing of the appropriate breakers, isolating the fault. It has become apparent that the close synergy of communications and the power grid will enable its fine-grained management, supporting the timely adaptation to increasingly dynamic operating conditions.

However, such applications come with stringent end-to-end communication delay requirements, i.e., in the order of a few tens of milliseconds [18, 99, 100]. In turn, the expected benefits from the envisioned cyber-physical system depend on the ability of the communication infrastructure to actually support these requirements. While high capacity optical fiber may be typically available on the transmission level (i.e., high voltage (HV) domain), adopting a similar approach on the distribution level (i.e., in the medium voltage (MV) domain) raises significant concerns with respect to the associated costs. Our analysis of a large set of real topologies (cf. Section 3.2.2) shows that the mostly urban environment of the distribution grid calls for a dense deployment of high capacity communication links, as opposed to the HV

[1]ADNs are defined as distribution networks that have systems in place to control a combination of distributed renewable energy resources like generators, loads and storage. Distribution network operators have the possibility of managing electricity flows via a flexible network topology.

domain [101]. As a result, the full-fledge fiber optic communication deployment in urban environment for MV distribution grid is currently not practical and plagued with various difficulties and prohibitive costs. Recent works have alternatively investigated the use of wireless technologies such as WiMAX and LTE [102], reporting however concerns about the impact of control plane and medium access control (MAC) layer delays, which is directly affected by the number of devices accessing the high capacity wireless channel(s) [18, 103, 104]. On the other hand, the readily available power line communication (PLC) infrastructure has relatively low costs, but the typically low PLC bandwidth appears as a bottleneck to the timely delivery of delay sensitive monitoring traffic. Based on the above observations, we identify the tradeoff between the performance gains from the deployment of high bandwidth technologies and the deployment costs (and/or MAC/signalling delay penalties in the case of wireless technologies) associated with wide scale PMU deployment in the MV domain.We highlight that this is the first work in MV domain investigating low latency communication infrastructure for PMU-based applications.

We address this tradeoff by considering the design of a hybrid communication infrastructure, where the existing PLC infrastructure is utilized to reduce the number of high capacity links required to satisfy the low latency requirements along with the associated costs. Our problem resembles a facility location problem, where we seek the minimum number and location of high-capacity links in the MV grid to satisfy our application-level latency constraints. As the problem is known to be NP-hard [105, 106], we turn our attention to heuristic-based solutions. To this end, and in order to guide the design of our solution, we engage in an in-depth analysis of the end-to-end delay (T_{e2e}) components. Based on a large set of 14 real MV grid topologies operated by a large DNO in the Netherlands, we perform an analysis of important topological characteristics of the MV domain [107], while also paying attention to PMU communication specificities such as the impact of precise PMU data synchronization. Our analysis yields valuable and pragmatic insights for the design of both low-cost and low-latency communication infrastructures for the MV grid, which we embody in the design of three different heuristic-based optimization algorithms. An extensive set of detailed packet level simulations demonstrate the effectiveness of our algorithms.

3.2 Background and Problem Statement

Designing a communication infrastructure for the support of a purpose specific cyber-physical system, such as the smart power grid, necessitates a good understanding of the operational context, in our case of the MV power grid. Fig. 3.1 provides a high level illustration of a typical MV grid, i.e., the (power) distribution network. A typical MV grid topology has a tree-like structure rooted at a primary-substations (P-SS), which is responsible for stepping down the transmission voltage from HV to MV. Each tree branch emanating from a P-SS corresponds to a distinct feeder (cable) further distributing the MV power to the desired areas through a series of secondary-substations (S-SSes), responsible for further stepping down the voltage. The power distribution network consists of multiple such trees rooted at different P-SSes.

3.2.1 Delay-Sensitive Synchrophasor Monitoring Applications

Our work is motivated by the challenge to support 3-phase RTSE application. RTSE is considered as an important tool for DNOs as it supports particularly important energy management and protection operations, such as fault detection/localization, post-fault management and voltage control [108, 109]. PMUs enable the support of such applications by monitoring power system parameters (e.g., phase angle, voltage, rate of change of frequency (ROCOF), etc.) at strategically selected S-SSes in the MV grid[2](see Fig 3.1). All PMUs are GPS-synchronized and stream their measurements to phasor data concentrators (PDCs), which are typically located at the P-SS. PDCs collect, time align and deliver synchrophasor data to applications such as RTSE.

Although typical refresh rates of state estimation processes are of the order of a few minutes, the high system dynamics of ADNs, due to renewable energy sources

[2]The selection of PMU locations constitutes a research area on its own (e.g., [110]). Without loss of generality, we consider a scenario with a PMU deployed at approximately every two S-SSes along a feeder (see Fig 3.1).

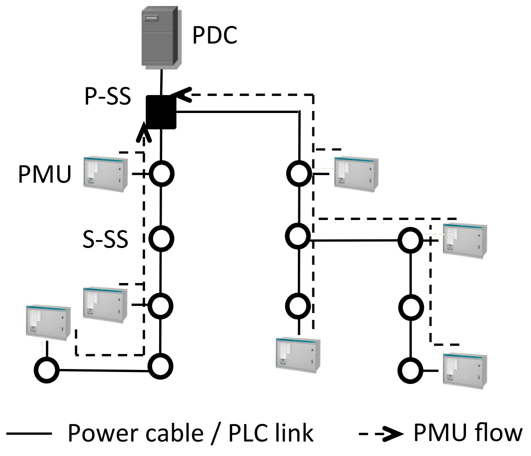


Figure 3.1: Medium voltage power grid.

(DRERs) and electric vehicles (EVs), necessitate the fine-grained estimation of system state within a few tens/hundreds of ms [98]. PMU reporting frequencies (F) of 50 or 60 frames-per-second facilitate this detailed view of the power grid [32]. Based on PMU data semantics [111], a realistic PMU message payload size is 102 bytes[3]. Further considering UDP and IP headers, and a 32-byte SHA-256 message authentication code, the overall data rate for each RTSE PMU flow delivered to the link layer is 64.8Kbps, for $F = 50$Hz.

The timely delivery of these measurements is a challenge for the underlying communication infrastructure. In this work, we account for RTSE applications a maximum total latency of 100ms [18, 98, 100], including latencies for PMU signal acquisition, PMU synchrophasor estimation and data encapsulation, communication network delay, PDC data frame time alignment, bad data detection and state estimation [100]. The time budget left for telecommunication network delay (T_{e2e}) depends on these latency components and has typically a constraint (denoted as T_{max}) of 20ms [19], at a PMU reporting rate of 50Hz [18, 98]. It was recently shown that the telecommunication network delay constraint could be further relaxed to 35-55ms due to new advancements in state estimation algorithms [100]. Nevertheless, in this work, we focus on $T_{max} = 20$ms, as a more stringent requirement[4]. At this

[3]Considering PHNMR=6, ANNMR=6 and DGNMR=2, with 32-bit floating-point accuracy [111].

[4]We note though that this is only an input parameter to the proposed algorithms (see Section 3.4),

point, it is important to stress that at the application level, RTSE necessitates the availability of *all* synchronized PMU measurements within the defined T_{max}. Otherwise, state estimation will suffer in terms of accuracy; hence, T_{max} stands for the worst case T_{e2e} acceptable.

3.2.2 Deployment Cost Minimization Problem

The support of the identified latency requirements depends heavily on the underlying communication infrastructure, which in turn is largely determined by the locations of the communicating entities and the selected transmission technology. We first consider a baseline communication network model enabled by PLC technologies [112], which, by allowing DNOs to make use of their existing power-line cables as the transmission medium, constitute the most straightforward and low-cost option for the support of communications in the power grid. In this baseline scenario, the communication network topology coincides with the MV power grid topology. We investigate the topological properties of the resulting communication network model based on a set of 14 MV power grid topologies operated by a DNO in the Netherlands. Table 3.1 summarizes the basic aggregated topological characteristics of the considered MV grids. Furthermore, in Table 3.2, we present the topological properties per area. Our dataset shows close agreement with literature (e.g., as surveyed in [21]) and thus, representative to general MV grids.

We represent the distribution grid, and the corresponding baseline communication network model, as a set of tree graphs, $G(V,E)$, with $v \in V$ as substations where node v_0 represents the root (i.e., the P-SS) [5]. The edges, $e \in E$, represent physical cables connecting different SSes. Then, we denote the distance in hop count between v_i and $v_{i'}$ as $d(v_i, v_{i'})$ with $i \neq i'$. Further, let U be the set of nodes (S-SSes) equipped with PMUs, comprising PMU-enabled nodes v_i^j, where $j \in [0..|U|-1]$ is the PMU index and $i \in [0..|V|-1]$ is the node index. We define $P(v_i^j)$ as the shortest path comprising the consecutive edges connecting PMU-enabled node, v_i^j, to v_0

not affecting their general applicability.

[5]On the communication level, v represent routing/switching devices located at the corresponding S-SSes, forwarding data packets.

Table 3.1: Summary of real MV grid topological properties of a large European DNO.

Primary Substations (P-SS)	14
Secondary Substations (S-SS)	1323
Number of edges (cables)	1426
Average cable length	498m
Average node degree	2.02

Table 3.2: Real MV grid topological properties per area.

Grid	Number of nodes	Number of edges	Mean node	Link[7] density degree	Mean[8] path length	Mean[9] betweenness
Area 1	187	223	2.0744	0.0128	7.3105	6.2774
Area 2	112	134	2.7077	0.0216	7.745	6.6869
Area 3	36	43	1.9545	0.0683	6.2778	5.1351
Area 4	155	177	2.1718	0.0148	7.1290	6.0897
Area 5	89	102	2.125	0.02604	7.1290	6.2247
Area 6	82	82	1.9759	0.02469	3.7195	2.6867
Area 7	22	22	1.9130	0.0952	3.4545	2.3478
Area 8	177	177	1.9887	0.01136	5.3728	4.3483
Area 9	28	28	1.9887	0.07407	5.7857	4.6207
Area 10	50	51	2	0.04163	4.5	3.4313
Area 11	101	101	1.9803	0.02	5.5049	4.4608
Area 12	98	98	1.9798	0.02061	4.55102	3.5152
Area 13	41	41	1.9524	0.05	2.5854	1.5476
Area 14	145	147	2.0119	0.0141	5.2897	4.2603

(see dashed lines in Fig. 3.1). The length of $P(v_i^j)$ is $|P(v_i^j)| = d(v_i^j, v_0) = d(v_i, v_0)$[6].

In the PLC-enabled baseline model, PMU flows (dashed arrows in Fig. 3.1) reach the PDC by traversing their uphill PLC links towards the root of the tree topology. Following the PMU deployment scheme described in Section 3.2.1, for the available MV grid topologies, we simulate the operation of 795 PMUs in a detailed packet-level simulation environment (see Section 3.5). Fig. 3.2 shows the cumulative fraction of the T_{e2e} observed at the PDC for a duration of 10 minutes with PLC bandwidth values: 100Kbps and 500Kbps [112][10]. The vast majority of PMU

[6]For clarity, for the rest of the paper, we simply refer $P(v_i^j)$ as P_j since there is only one unique path from a PMU to the PDC.

[7]Path length represents the number of hops from a S-SS to the P-SS.

[8]Link density=$\frac{|E|}{(|V|-1)*|V|/2}$.

[9]Betweenness represents the number of shortest paths between a S-SS and the P-SS that involve the measured node.

[10]PLC encompasses a diverse set of technical realizations with different bandwidth values, broad-

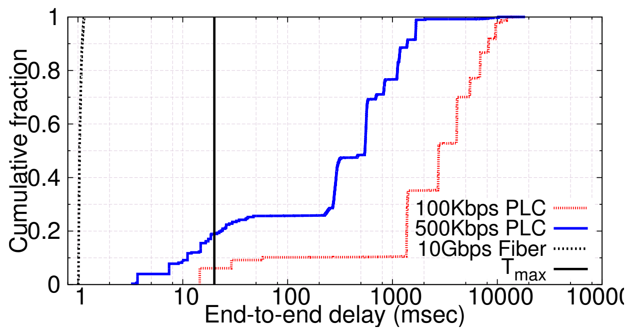


Figure 3.2: (CDF) T_{e2e} of PMU flows with PLC and optical fiber

messages delivered exceeds T_{max}. Clearly, the considered set of applications cannot be supported by PLC technology alone. However, we will show in Section 3.3 that limited bandwidth is not the only key delay factor.

We further consider and simulate an optical-fiber based communication network model, following the current practice in HV deployments [101]. In particular, we consider 10Gbps optical fiber links directly connecting PMU-enabled S-SSes to the PDC at P-SS. As shown in Fig. 3.2, this communication infrastructure fully conforms to the T_{max} constraint. However, it necessitates the deployment of 795 optical fiber links in total, representing a significant CAPEX.

Recent studies have also shown that the adoption of wireless technologies may lead to an increase of medium access delays due to the contention for access to the shared wireless medium, even in cases where no other background traffic is served [18, 103, 104]. This contention and the corresponding delays increase with the number of wireless transmitting devices, i.e., subject to the selected wireless technology, an increased volume of attempts to transmit increases the collision probability, leading to back-off/scheduling delays. Given that existing wireless networks (e.g., cellular (A-)LTE, WiMAX) have been dimensioned for a particular access load, the introduction of additional devices (i.e., PMUs) raises concerns about the aforementioned performance penalties. Of course, increasing frequency reuse with the deployment of smaller cells would reduce contention, for a certain access demand. However, this would obviously come at a significant deployment

band 500Kbps being one of them. Our methodology can be applied for different bandwidth values. For extremely low values, lower datarate PMU configurations should obviously be considered.

cost for communication network operators[11]. The synchronization of PMUs only further exacerbates the contention issue, since it increases collision probabilities and/or limits scheduling flexibility. For all these reasons, it follows that the number of wireless transmitting devices should also be kept to a minimum.

In short, PLC, though readily available, appears unable to support the considered low latency applications, urging for alternative solutions such as the use of modern wireless or high-speed wired technologies. However, the deployment of such technologies incurs a non-negligible capital expenditure and/or performance penalties. In this respect, it becomes apparent that the scale of deployment of high capacity links needs to be carefully considered. Considering this tradeoff between deployment costs and performance, we propose the design of hybrid communication infrastructures that exploit the existing low cost PLC capabilities, while also employing higher bandwidth technologies. The rationale is to take advantage of the availability of PLC to *partially* accomplish the task of delivering the PMU data flows to the PDC, thus reducing the number of high capacity links in the overall network. Starting from our baseline network model, the objective then becomes to select the minimum sub-set of S-SSes to be equipped with high capacity direct links to the P-SS/PDC (e.g., optical fiber) and act as *sink* nodes, i.e., aggregating PMU traffic through PLC links. The envisioned scenario is illustrated in Fig. 3.3.

Let X_i with $i \in [0..|V|-1]$ be a binary decision variable, set to 1 if node v_i is equipped with a high capacity communication link; we denote such a node with v_i^k with $k \in [0..|S|-1]$, where S be the set of sink nodes. Let also Y_{jk} with $j \in [0..|U|-1]$ and $k \in [0..|S|-1]$ be a binary variable set to 1 if a PMU flow from v_i^j is delivered to a sink node $v_{i'}^k$[12]. Then, denoting the end-to-end delay of each PMU flow over path P_j (with $j \in [0..|U|-1]$), as T_{e2e}^j, our objective can be loosely expressed as follows:

[11]Dedicated, private wireless networks constitute another option for DNOs. However, they are associated with other types of deployment costs, e.g., spectrum licence costs. We consider this particular aspect out of the scope of this paper.

[12]Note that $i = i'$ is allowed i.e., a S-SS can be equipped with both a PMU and a high capacity link.

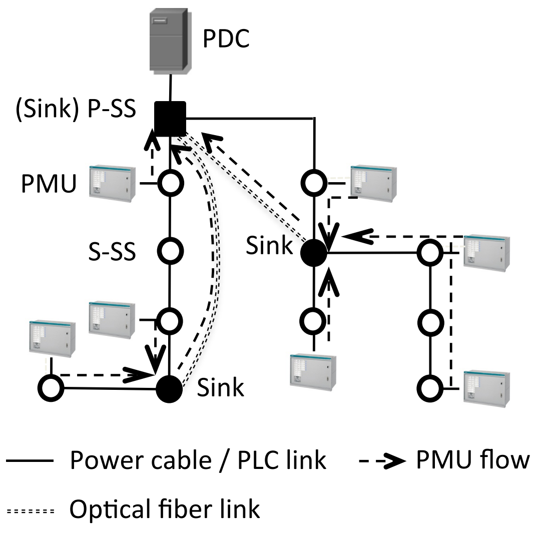


Figure 3.3: Hybrid communication infrastructure.

$$
\begin{aligned}
&\text{minimize} && \sum_i X_i, && && (3.1)\\
&\text{subject to} && T_{e2e}^{P_j} \le T_{max}, && \forall j \in [0..|U|-1] && (3.2)\\
& && \sum_k Y_{jk} = 1, && \forall j \in [0..|U|-1], && (3.3)\\
& && && k \in [0..|S|-1] && \\
& && X_i \in \{0,1\}, && \forall i \in [0..|V|-1] && (3.4)\\
& && Y_{jk} \in \{0,1\}, && \forall j,k && (3.5)
\end{aligned}
$$

The exact nature of the problem and the corresponding solution obviously depend on the first constraint which only roughly expresses the low latency requirement. The second constraint ensures that each PMU-enabled node sends its flow to a single sink node. To assess the hardness of our problem, we can merely express the first constraint by setting an upper limit (i.e., d_{max}) for the distance between a PMU-enabled node v_i^j and the corresponding sink location $v_{i'}^k$. This results in constraint (1) to be re-written as follows (see also Section 3.3.2.1):

$$d(v_i^j, v_{i'}^k) \le d_{max}$$

Even in this simple case, the resulting problem is a typical NP-hard *facility location* optimization problem [105, 106] thus turning our attention to heuristic-based solutions. To further explore the problem space and guide the design of our heuristics, we first decompose T_{e2e} into its constituents and investigate the most important factors impacting them (Section 3.3). In this effort, we get valuable input from the detailed investigation of our large set of MV topologies. Our analysis yields important insights for the subsequent design of the proposed heuristic algorithms (Section 3.4).

3.3 Analysis of End-to-End Delay Impact Factors

Our analysis of the various latency impact factors is enabled by the identification of the various components of T_{e2e}, i.e.,

- Processing delay ($proc$): the time used for operations such as medium adaptation, (de)coding, switching, routing, message authentications codes generation / verification.
- Propagation delay ($prop$): depends on the transmission medium and the distance travelled by the signal. For copper cable, this is typically 5ns per meter.
- Transmission delay ($trans$): the time required to transmit the data and is subject to the bandwidth of the underlying transmission technology.
- Queuing delay ($queue$): the time spent by data waiting for transmission at the transmitting devices.

We consider for this analysis a discrete time domain divided into slots with each slot capable of containing exactly one PMU packet. For each delay component $x \in \{proc, prop, trans, queue\}$, we consider the corresponding per hop delay t_x. Additionally, we define the aggregate T_x of each delay component x over a path P_j as $T_x = \sum^{|P_j|} t_x$.

In the following subsections we investigate the impact of the key factors affecting the aforementioned delay components in order to get insights on where to place high capacity links to achieve the low latency requirement in a cost-efficient manner.

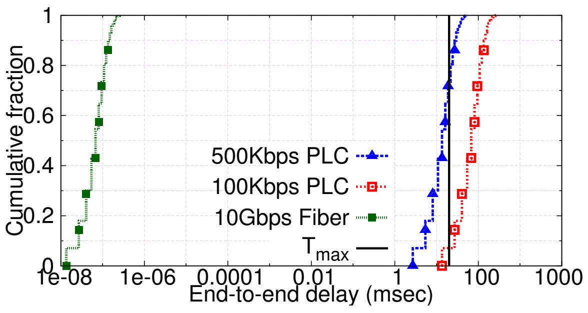
3.3.1 Bandwidth

Bandwidth availability impacts both t_{trans}/T_{trans} and t_{queue}/T_{queue}. Obviously, t_{trans}/T_{trans} increase with lower bandwidth values. Moreover, queuing delays perceived at a node increase when the available output bandwidth is lower than the incoming data rate at the node [13]. Fig. 3.4(a) shows the cumulative fraction of the T_{trans} across all PMU-to-PDC paths, for the cases of PLC and optical fiber based communication infrastructures. For the PLC-based case, T_{trans} exceeds T_{max} for 92.91% and 26.59% of the transmitted packets for the cases of 100Kbps and 500Kbps respectively. In the case of optical fiber, we see a considerable reduction of accumulated T_{trans} compared to the PLC case, leaving abundant delay budgets for other delay components. This is a direct consequence of the reduction of t_{trans} values from $13.52ms$ or $2.70ms$ for 100Kbps and 500Kbps PLC datarates, respectively, to only $t_{trans} = 13.52\mu$s for the case of optical fiber (for the considered payload size and header overheads; see Section 3.2.1).

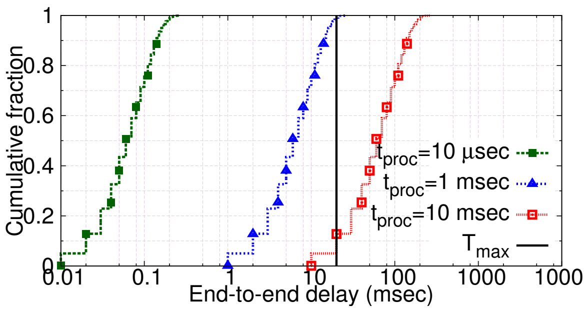
Our simulations for the baseline PLC scenario (Section 4.2.2) also indicate that on average, T_{queue} accounts for 96.88% of T_{e2e}, with T_{trans} and T_{proc} accounting only for 2.18% and 0.94% respectively[14]. This domination of T_{queue} on T_{e2e} implies the lack of sufficient bandwidth to support the PMU traffic. Although the perceived T_{queue} and T_{trans} evidently demonstrate the role of the adopted technology's bandwidth, they are still dependent on a series of other factors including the communication network topology and the tight synchronization of PMUs. We further investigate these aspects next.

[13]As queuing delays are also related to both topological aspects of the communication network and the synchronization of PMUs, we discuss them in detail in Section 3.3.2

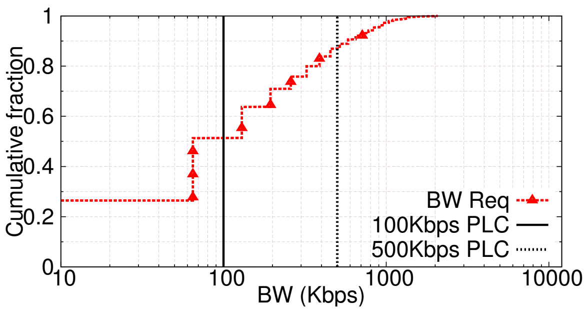
[14]Due to the short distances between S-SSes (see Table 3.1), we omit t_{prop}/T_{prop} in the following as it is only in the order of microseconds.



(a) (CDF) T_{trans} when using various technologies and t_{trans}



(b) (CDF) T_{proc} when using purely PLC technology with various t_{proc}



(c) (CDF) PMU bandwidth requirements; traffic volume exceeds PLC capabilities

Figure 3.4: Impact of topology on PMU application performance

3.3.2 Topology

3.3.2.1 Path length

The path length, $|P_j|$, has an important impact on perceived aggregate T_x latencies, since lengthy paths accumulate delays on multiple hops. We further use our set of MV topologies to realistically quantify this impact. Fig. 3.4(b) shows the cumulative fraction of the processing delays accumulated by data packets across all PMU-to-PDC PLC paths (i.e., $P_j : \forall j \in [0, \ldots, |U|-1]$), for a range of per node processing delay values, t_{proc}. These values depend on the computational resources of the forwarding devices and can vary significantly, ranging from a few micro-seconds to even milliseconds per packet [113]. If we consider recent overlay approaches [33, 100], these delays may further increase due to the transition of packets from the kernel to the user space. We notice that, subject to t_{proc}, the overall delay penalty T_{proc} may get close or even exceed T_{max}, even though T_{proc} constitutes only 0.94% of T_{e2e} (for $t_{proc} = 1ms$). Similarly, as previously discussed, Fig. 3.4(a) shows T_{trans} values close to T_{max}, though T_{trans} constitutes only 2.18% of T_{e2e}. This is a direct effect of path lengths, which in our topologies have an average and maximum value of 5.84 and 20 hops respectively.

In essence, these measurements yield an important guideline for the design of low latency communication networks: in the presence of high t_{proc} values (i.e., in the order of 1ms), bandwidth availability alone may not suffice in keeping T_{e2e} values low, when paths are considerably long e.g., interconnecting P/S-SS with optical fiber, following the power grid topology. Moreover, the provisioning of computational resources at each forwarding node should be carefully considered.

Building on these observations, we re-formulate the first constraint of our optimization problem (Eq. 3.1). Namely, to limit the effect of path lengths on T_{e2e}, we constrain the maximum number of PLC hops by limiting the distance between a PMU and its sink node (d_{max})[15]:

$$d(v_i^j, v_{i'}^k) \leq d_{max} = \left\lfloor \frac{T_{max} - t_{proc}}{t_{trans} + t_{proc}} \right\rfloor, \tag{3.6}$$
$$\forall i \in [0..|V|-1], \forall j \in [0..|U|-1], \forall k \in [0..|S|-1]$$

For the cases of 100Kbps and 500Kbps PLC, we get $d_{max} = 1$ and 5 hops respectively, as d_{max}'s limit values.

[15] t_{prop} (average $\leq 3\mu$s in the considered topologies) and t_{trans} on the sink-to-PDC link ($\leq 2\mu$s for a 10Gbps optical fiber link) are considered negligible. However, we account the t_{proc} for the sink-to-PDC hop.

3.3.2.2 Application-level betweenness

As previously mentioned, T_{queue} constitutes 96.88% of T_{e2e}. This delay component depends on the relation between the available and the required bandwidth at each forwarding device. While the former depends on the selected transmission technology, the latter depends on topological aspects of the communication network. Fig. 3.4(c) shows the cumulative fraction of the total PMU traffic volume aggregated at each PLC link towards the PDC in our MV topologies. Again, we see that a PLC-based infrastructure fails to accommodate the resource requirements as for more than half of the communication nodes, the bandwidth requirements exceed a typical bandwidth value of 100Kbps ($\approx$ 10% for 500Kbps links).

To better understand this aspect, we introduce the concept of *application-level betweenness*, $b(v_i)$, as the number of shortest paths P_j crossing node, v_i. Note that $b(v_i)$ is determined both by the topology structure and the placement of the PMUs. In the considered set of MV grid topologies, we observe an average and maximum $b(v_i)$ value of 3.24 and 32 respectively. Considering a 64.8Kbps data rate per PMU flow, it is easy to understand the domination of T_{queue} in T_{e2e}.

Building on this observation, we formulate the next constraint for the design of our hybrid communication network topologies, i.e., we impose an upper bound on application-level betweenness (b_{max}) throughout the topology:

$$b(v_i) \leq b_{max} = \left\lfloor \frac{BW}{DR} \right\rfloor, \forall i \in [0..|V|-1] \tag{3.7}$$

where BW is the available PLC bandwidth and data rate (DR), $DR = 64.8Kbps$. For BW = 100Kbps and 500Kbps, this yields $b_{max} = 1$ and 7 respectively, significantly lower than the observed $b(v_i)$ values in the baseline network model.

3.3.3 Flow Synchronization

Another factor with significant impact on the T_{e2e} is the synchronized nature of PMU flows[16] whereby a PMU flow is defined as continuous traffic flow from PMU

[16]It is worth noting that the synchronization issue did not draw much attention in the HV domain because of the low PMU deployment density and the high bandwidth of the adopted transmission technologies [101].

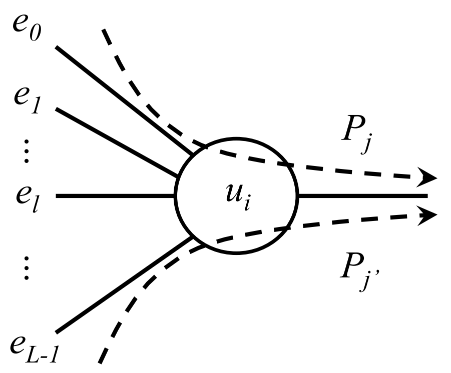


Figure 3.5: Example of path $P_{j'}$ (flow j') joining path P_j (flow j) at node u_i.

to PDC. As briefly mentioned in Section 4.2.2, such synchronization may significantly impact the delays for access to the wireless medium. However, PMU synchronization also has an important impact on the baseline PLC network model. Packets originating at different PMUs reach the same forwarding device at (almost) the same time. Consequently, packets wait in the transmission queue for a time linear to t_{trans}, i.e., waiting until all interfering packets from other PMU(s) get transmitted. Our simulation results show that approximately 20% of PMU flows experience such synchronization problem across 12.45% of forwarding nodes, inflating the overall observed T_{queue}.

To assess the impact of synchronization, we follow the approach proposed in [114]. Specifically, we focus on the worst-case scenario, i.e., a packet has to wait for all other packets (almost) simultaneously arriving the same node, to get transmitted first. We consider this worst-case scenario as our target is to limit the maximum T_{e2e} perceived.

We focus on a node of interest, v, with L inbound links $e_l, l \in [0..|L|-1]$ and e_f outbound link. Further, a path, $P_{j'}$, is said to join path P_j when they share the same outgoing edge e_f but not an incoming edge e_l at the node of interest (see Fig. 3.5). Let $R_{P_j}(e_f)$ be the number of paths, $P_{j'}$, that join $P_j : \forall j' \neq j; j', j \in [0..|U|-1]$, at edge e_f. Then, the *Route Interference Number* (RIN) of path P_j is defined as follows:

$$R(P_j) = \sum_{e_f \in P_j} R_{P_j}(e_f).$$

By counting the number of interfering paths at each forwarding node towards the PDC, RIN allows us to derive the maximum number of times a PMU packet can be delayed due to synchronization in the case where all PMUs send a single packet. In this case, [114] showed that the overall end-to-end T_{queue} of the packet sent on P_j is bounded by $R(P_j)$.

When the b_{max} constraint is met, the aforementioned single packet case can be generalized into a multi-packet case where PMUs send one packet at each measurement interval, $1/F$. This generalization is possible because measurements taken at one interval will only arrive after all measurement packets from preceding intervals have been transmitted. Furthermore, when the number of flows of each link e_f is lower than the maximum $b(v_i) : \forall i \in [0..|V|-1]$, the worst-case queueing delay of path P_j, $T^{P_j}_{queue}$, is bounded by a tighter upper bound compared to RIN [115, 116]. To state this delay bound, let $\beta(e_f)$ denote the number of interfering packets at edge, e_f and Q_l denote the number of paths from inbound edge e_l. Then we express $\beta(e_f)$ and the corresponding worst-case queueing delay, $T^{P_j}_{queue}$, as follows:

$$\beta(e_f) = \sum_l Q_l - \max_l\{Q_l\} \tag{3.8}$$

$$T^{P_j}_{queue} = \sum_{e_f \in P_j} \beta(e_f) t_{trans} \tag{3.9}$$

where function $\max\{Q_l\}$ selects at the outbound edge, e_f, the maximum number of Q_l from all inbound, e_l.

We can then extend the notion of worst-case queueing delay bound to include t_{proc} and t_{trans} along the path to the PDC. Then, $T^{P_j}_{e2e}$ is calculated as follows:

$$T^{P_j}_{e2e} = \Sigma_{e_f \in P_j}\left\{\beta(e_f)t_{trans} + t_{trans} + t_{proc}\right\} \tag{3.10}$$

Based on this formulation, we take into account synchronization when satisfying the constraint:

$$T_{e2e}^{P_j} < T_{max} \tag{3.11}$$

3.4 Design of Network Planning Algorithms

Building on the identified constraints, we next describe three heuristic-based algorithms for the design of low latency and low cost hybrid communication infrastructures. Each algorithm is tailored for specific application environments.

- The *path length constraint (PLeC) algorithm* selects sink node locations by constraining the length of data delivery paths (with d_{max}, Eq. 3.6), so that the accumulated T_{proc}, T_{prop} and T_{trans} are also capped (see Section 3.3). Since it does not cater for bandwidth availability, this algorithm is most suitable for low DR applications (e.g., low DR PMU reporting) and can be employed for environments where multiple (low DR) applications share the same communication infrastructure.

- The *application-level betweenness and path length constraint (AB-PLeC) algorithm* selects the sink locations by constraining both path lengths and the number of PMU flows on each PLC link (with b_{max}, Eq. 3.7); therefore, explicitly targeting the reduction of T_{queue}. By adjusting the b_{max} constraint according to the residual bandwidth of each link, AB-PLeC can be easily adapted to cater for background traffic, i.e., from applications expected to share the same communication infrastructure (e.g., Intelligent Electronic Device based monitoring).

- The *flow interference and bandwidth constraint (FIB) algorithm* selects the sink locations by explicitly seeking the nodes at which a PMU packet exceeds T_{max} in the worst-case scenario (see Eq. 3.11), limiting both b_{max} and $\beta(e_f)$ values. In contrast to the first two algorithms, FIB takes synchronization into account; however it is tailored for cases of dedicated communication infrastructure, i.e., no background traffic.

Algorithm 1 PLeC algorithm

```
Input: $G, d_{max}$
Output: S
1: $S \leftarrow \emptyset$
2: for all $i$ do
3:     $a_i \leftarrow d_{max}$
4: end for
5: while $G \neq \emptyset$ do
6:     $v_l \leftarrow G.getRandomLeafNode()$
7:     if $v_t \neq v_0$ then
8:         $v_p \leftarrow v_l.getParentNode()$
9:         $a_p \leftarrow min(a_p, a_l - 1)$
10:        $G.removeNode(v_l)$
11:        if $a_p = 0$ then
12:            AddSink($G, S, v_p$)
13:        end if
14:    else
15:        AddSink($G, S, v_l$)
16:    end if
17: end while
18: return $S$,$M$
19:
20: function AddSink($G, S, v_s$)
21:     $S \leftarrow S \cup v_s$
22:     $R \leftarrow \emptyset$
23:     for all $v_i \in G$ do
24:         if $d(u_i, u_s) \leq a_i$ then
25:             $M_s \leftarrow M_s \cup v_i$
26:             $R \leftarrow R \cup v_i$
27:         end if
28:     end for
29:     $T \leftarrow G \setminus R$
30: end function
```

3.4.1 Algorithm Based on Path Length Constraint

For the PLeC algorithm, we follow the distance constraint formulation of the *p*-center facility location problem [106]. We define $S = s_1, \ldots, s_m$ as the set of sink nodes, with $1 \leq m \leq |V|$. Further, let $D(S, v_i) = min\{d(s, v_i) : s \in S\}$, the distance between each node v_i and its nearest sink node. Our objective is to find the minimum set S such that for all $D(S, v_i) \leq d_{max}$. We solve this problem via the *sequential location procedure* proposed in [106]. Our algorithm (see Algorithm 1) takes as input the tree topology, G and the distance constraint, d_{max}, and outputs the set of selected sink nodes, S, along with set M (see next). For all nodes v_i, we define a distance value $a_i, i \in [0..|V| - 1]$ and a set M_i, which contains the nodes that can use node v_i as their sink node, under the d_{max} constraint. We further set $M = \bigcup_{i \in [0..|V|-1]} M_i$.

The algorithm starts by randomly selecting a leaf node, v_l from G, along with its parent node v_p. Traversing the tree hierarchy towards its root, the algorithm updates the distance value a_p of nodes v_p as in line 9, until it reaches 0. Note that the hierarchy is traversed by removing the visited leaf nodes from the topology. When $a_p = 0$, node v_p is added to the sink node set (function $\textsc{AddSink}(G, S, v_m)$, line 20). In this step, all nodes v_i whose minimum hop distance to the new sink v_m is below their a_i value are added to the M_m set. All nodes assigned to the new sink are also removed from the tree[17].

The outcome of the algorithm consists of the sets M_i for each selected sink node v_i. These sets may overlap with each other in cases where more than one sink nodes reside within the d_{max} range of some node. At the same time, subject to the exact topological characteristics of tree G, sets M_i may not all have the same size. This means that a careless assignment of nodes to sinks may result in the overloading of some sink nodes both with respect to their processing and bandwidth capabilities. We address this through a simple node assignment procedure which balances the load between sink nodes. Based on the available M sets, the procedure first produces sets L_i which hold the set of all sink nodes within d_{max} range of each node v_i. The members of each L_i set are ordered in increasing hop distance to v_i. The sink node at the smallest distance is selected. When multiple sink nodes are located at the same distance, the algorithm selects the preferred sink node v_m with the minimum M_m size so as to not overload other sinks which can possibly serve more nodes.

3.4.2 Algorithm Based on Application-Level Betweenness and Path Length Constraint

The AB-PLeC algorithm finds the set of sink locations that constrains the number of PMU flows being forwarded by each PLC link while maintaining the d_{max} constraint. It takes as input the tree graph topology G, d_{max}, b_{max} and the maximum number of PMU flows that can be accommodated by a high bandwidth link connecting a sink node to the PDC, b'_{max}. b'_{max} is set in a similar way to b_{max}, considering the available high capacity link bandwidth value, and it is therefore normally ex-

[17]This process may result in a forest. Structure G is used for all trees, and `getRandomLeafNode()` (line 6) returns a leaf node randomly selected from any of the trees.

Algorithm 2 AB-PLeC algorithm

```
Input: $G, d_{max}, b_{max}, b'_{max}$
Output: S
 1: $S \leftarrow \emptyset$
 2: for all $i$ do
 3:     $a_i \leftarrow d_{max}$
 4:     $b_i \leftarrow (v_i.hasPMU())?1 : 0$
 5: end for
 6: while $G \neq \emptyset$ do
 7:     $v_l \leftarrow G.getRandomLeafNode()$
 8:     $v_p \leftarrow v_l.getParentNode()$
 9:     $x \leftarrow min(a_p, a_l - 1)$
10:     $y \leftarrow b_p + b_l$
11:     if $v_l.markedAsSink()$ then
12:         $S \leftarrow S \cup v_l$
13:         $G.removeNode(v_l)$
14:     else
15:         if $y > b'_{max}$ and $v_p \neq v_0$ then
16:             $S \leftarrow S \cup v_p$
17:             $G.removeNode(v_p)$
18:         else
19:             $a_p \leftarrow x$
20:             $b_p \leftarrow y$
21:             $G.removeNode(v_l)$
22:             if $b_p > b_{max}$ or $a_p \leq 0$ then
23:                 $v_l.markAsSink()$
24:             end if
25:         end if
26:     end if
27: end while
28: return $S$
```

pected to be considerably higher than b_{max}. In addition to a_i, for each node v_i, we define b_i as the current $b(v_i)$. All b_i values are initialized to 0, unless a PMU is attached to the corresponding node (line 4). The tree topology is traversed from the leafs towards the root node, allowing the forwarding of PMU flows over PLC links up to the point where the uplink capacity of a visited node is exceeded (line 22). This node is then selected to act as a sink location (line 23). PMU flows from additional descendants in the tree may be added, subject to the b'_{max} value (line 15). Visited nodes and sinks are removed from G and the algorithm terminates when all nodes have been removed. Then, each node in the tree can forward its traffic to its closest ancestor sink node.

Algorithm 3 FIB algorithm

```
Input: $G, T_{treshold}, b_{max}, t_{trans}, t_{proc}$
Output: S
 1: $S \leftarrow \emptyset$
 2: for all $v_i$ in $G$ do
 3:     $U \leftarrow \cup(v_i.hasPMU())?v_i : 0$
 4: end for
 5: while $U \neq \emptyset$ do
 6:     $Hotspot \leftarrow \emptyset$
 7:     for all $i \in [0..|V|-1]$ do
 8:         $b_i \leftarrow calculateApplicationBetweenness()$
 9:     end for
10:     $U.disableHotspotLabel()$
11:     for all $v_i^j$ in $U$ do
12:         $T_j \leftarrow 0$
13:         $P_j \leftarrow u_j^i.getShortestPathToPDC()$
14:         for all $v_m$ in $P_j$ do
15:             $T_j \leftarrow T_j + WorstCaseDelayAt(v_m)$
16:             if $T_j \geq T_{max}$ or $check(b_{max})$ then
17:                 $v_m.markAsHotspo()$
18:                 $Hotspots \leftarrow Hotspots \cup v_m$
19:                 $Break$
20:             end if
21:             if $v_m.isHotspot()$ then
22:                 $Break$
23:             end if
24:         end for
25:     end for
26:     $leafhotspot \leftarrow Hotspots.getLeafHotspot()$
27:     $S \leftarrow S \cup leafhotspot$
28:     $U.removeChildrenPMUs(leafhotspot)$
29: end while
30: return $S$
```

3.4.3 Algorithm Based on Flow Interference and Bandwidth Constraint

Based on the delay bound formulation (Eq. 3.10), we propose a heuristic algorithm that constrains b_{max} and the number of interfering packets of each flow (via T_{max}), precisely identifying the required sink locations.

The algorithm takes as input the tree topology G, b_{max}, T_{max}, t_{trans} and t_{proc} and outputs the set S of sink node locations. The algorithm first creates a set U of the nodes equipped with a PMU and computes b_i for all $i \in [0..|V|-1]$ (line 8). In the second stage, for each PMU-enabled node v_i^j, FIB parses G towards the PDC accumulating the worst-case delay at each node (line 15). When either the calculated delay at a node v_m in P_j reaches T_{max} or b_{max} is violated, the algorithm marks v_m as a hotspot. After parsing all nodes in U, the FIB algorithm finds the hotspot of each PMU flow. In the third stage, the algorithm selects a *leaf* hotspot (i.e., a hotspot with no hotspot descendants) that is farthest to v_0 and adds it into the sink set, S. In the fourth stage, FIB removes the sub-tree rooted at the selected sink location from G. The above four stages are repeated until all PMU-enabled nodes have been removed from U.

3.5 Performance Evaluation

We apply the proposed algorithms on the available tree-like MV power grid topologies and derive a series of alternative communication network topologies under specific constraints. Based on the derived topologies, we perform an extensive set of detailed packet level simulations. We focus on the case of 500Kbps but similar conclusions apply for the case of 100Kbps. We consider each sink node to be connected to the P-SS with a 10Gbps optical fiber link and set $t_{proc} = 1$ms. Based on the above, we then get $d_{max} = 5$, $b_{max} = 7$ and $b'_{max} = 147$ as the topological metrics that would conform to the desired T_{e2e} requirement. Table 3.3 summarizes the results for the various derived topologies. We denote the constraints considered by each algorithm as PleC(d_{max}) and AB-PLeC(d_{max}, b_{max}). For each topology, we show the percentage of packets measured to exceed T_{max}, the maximum T_{e2e}, the total number of sink node locations, i.e., the number of high capacity links required, and the gain in terms of the reduction percentage of fiber links compared to the full optical fiber scheme. Figures 3.6 and 3.7 further show the cumulative fraction of T_{e2e} of all packets, for the various topologies.

We see that FIB, PleC(2) and AB-PLeC(3,7) fully satisfy the delay constraint while requiring only 163, 309 and 256 sink nodes respectively. This constitutes a reduction in the order of up to 80% compared to the case of ubiquitous optical fiber deployment, requiring 795 such links. PLeC(2) achieves an overall better performance with median and maximum delay values of 4.7ms and 14.8ms, against 7.4ms and 20ms of AB-PLeC(3,7) respectively. AB-PLeC(4,7) and PLeC(3) closely follow, only slightly exceeding T_{max} for $< 1\%$ of the measured packets, i.e., by 2.9ms

Table 3.3: Summary of resulting topologies.

Sink deployment	% packets > T_{max}	max T_{e2e}(ms)	# sink nodes	% gain
PleC(2)	0%	14.8	309	61.13%
PleC(3)	0.25%	22	236	70.31%
PleC(4)	2.16%	25.6	188	76.35%
PleC(5)	15.94%	290	160	79.87%
AB-PLeC(3,7)	0%	17.5	256	67.79%
AB-PLeC(4,7)	0.67%	22.9	194	75.59%
AB-PLeC(5,7)	20.51%	39.1	147	81.5%
FIB	0%	19.56	163	79.49%
Full optical fiber	0%	1.14	795	0%

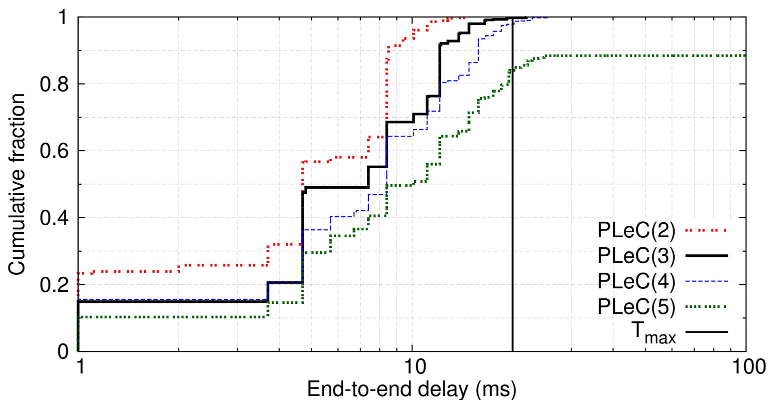


Figure 3.6: CDF of T_{e2e} for 500Kbps PLC links: PLeC algorithm

and 2ms respectively. Also, we see that PLeC(4) and AB-PLeC(5,7) achieve a maximum delay value of 25.6ms and 39.1ms. As discussed in Section 3.2.1, these latencies could be acceptable in cases of improved delay budgets [100], lowering the number of sink nodes to 194 and 236 respectively, i.e., an improvement in the other delay components could reduce the high capacity links by approximately 24% and 23% respectively.

PLeC(5), PLeC(4), AB-PLeC(4,7) and AB-PLeC(5,7) exceed T_{max}, even though we enforce the d_{max} and b_{max} constraint values derived from the considered MV topologies. In the case of PLeC(5), T_{e2e} reaches a maximum of 290ms. This is because the PLeC algorithm does not take into account the b_{max} constraint. Indeed, $b(u_i)$ values (for non-sink nodes) in PLeC(5) topologies reach a maximum value of 15, resulting in overloaded uplinks. However, this does not hold for AB-PLeC.

For AB-PLeC(4,7) and AB-PLeC(5,7) the non-conformance is attributed to

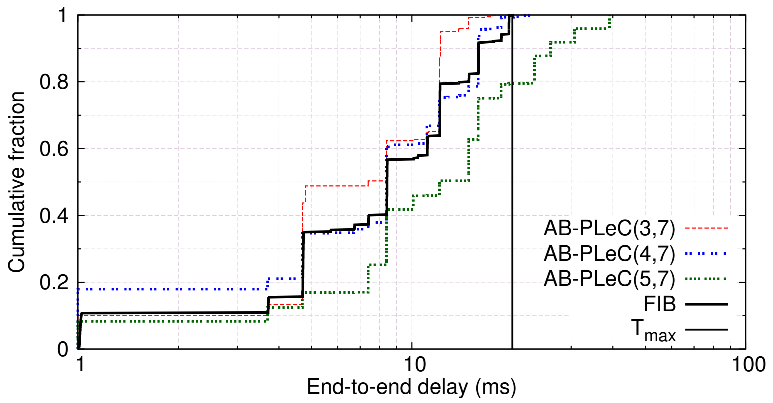


Figure 3.7: CDF of T_{e2e} for 500Kbps PLC links: AB-PLeC and FIB algorithm

PMU synchronization. Fig. 3.8 shows for each flow the relation between the length of the corresponding path to the PDC and the number of times the flow *may*[18] suffer synchronization events, i.e., its packets arrive at a node (almost) at the same time with packets of other flows[19]. Topology AB-PLeC(4,7) allows a maximum of 4 hops to a sink node for all PMU flows (hence 5 to the PDC), which leads to a delay of 15.82ms including $d_{max}(t_{trans} + t_{proc})$ from PMUs to sinks and a $t_{proc} + t_{trans_{10Gbps}}$ from sinks to the PDC. This leaves 4.184ms as the remaining budget for T_{max}. Given this time budget, the maximum number of t_{trans} a packet could afford to wait in the queue due to synchronization in AB-PLeC(4,7) is therefore 1 (i.e., 2.704ms). However, we observe that for AB-PLeC(4,7), some flows may experience synchronisation delays twice, thus exceeding T_{max}.

In contrast, the FIB algorithm presents the advantage of explicitly and precisely identifying the locations where T_{max} is reached. Compared to AB-PLeC(4,7), we see that FIB may yield even longer paths than AB-PLeC, however only for cases of limited synchronization events. For instance, Fig. 3.8 shows a 6-hop path with only one synchronization event. In essence, FIB postpones the selection of a sink location as much as possible, leading to sink nodes closer to the PDC, i.e., utilizing PLC as much as possible. In contrast, AB-PLeC(3,7) constrains the number of hops to sinks to 3 (4 hops to PDC), forcing packets that could still use PLC, to use

[18]Our analysis in Section 3.3.3 focuses on the worst-case scenario, which is experienced by only one of the flows.

[19]Obviously, multiple data points coincide in each case.

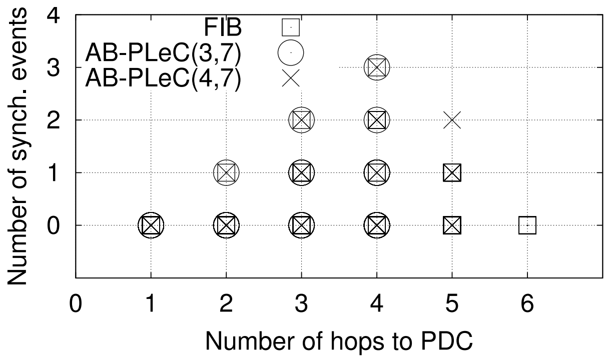


Figure 3.8: Each point corresponds to one flow and denotes the length of the path traversed towards the PDC and the number of times the flow may encounter synchronization delays.

the high capacity links of sink nodes. As a result a higher number of sink nodes must be unnecessarily deployed, i.e., a 56% increase of sink nodes against the FIB algorithm.

3.6 Conclusion

This chapter investigates the network deployment cost minimization problem for low latency applications. In particular, a case study in smart grid is carried out, whereby low bandwidth communication infrastructures in power grid need to be upgraded to a low latency smart grid. To this end, high bandwidth network links such as optical fiber need to be deployed on top of existing communication infrastructures. Therefore, the deployment cost minimization problem is to determine the minimum deployment cost for the added high bandwidth network links, so that the upgraded infrastructures can achieve the required low latencies.

To solve this problem, we first derive a set of practical guidelines for the design of low latency communication infrastructures, through a detailed study of the available topologies. Our investigation explicitly identifies, quantifies and addresses the effect of flow synchronization, which could largely affect end-to-end latencies. Building on our empirical observations, we propose and evaluate three heuristic algorithms that identify the locations in a given grid that should be equipped with high capacity links, striking a balance between low latencies and deployment costs. Enforcing our algorithms on the available MV topologies and additionally engaging in extensive packet-level simulations, we show that the proposed algorithms can indeed satisfy the targeted low latencies while reducing the extent of high capacity link deployment by up to 80% in comparison to ubiquitous deployment of direct interconnection between senders and receivers. We believe the proposed network planning algorithms can help power grid operators to achieve cost-effective transition towards low latency smart grid infrastructures. At the same time, the proposed algorithms can be easily applied to more general network upgrade cases, which will help ISPs to reduce deployment cost while achieving the required low latencies in the upcoming 5G era.

Chapter 4

Dynamic Resource Management for Low Latency Applications

In order to reduce ISPs' costs and provide flexibilities in new service deployment, NFV [117] was proposed to enable fully virtualized networks for ISPs. However, the question of how to dynamically manage different network resources in a cost-efficient manner for low latency communications has not been addressed. This problem involves dynamically finding *the optimal routing paths between users and end services*, and deriving *the optimal locations and amount of computing resources for end services*. In particular, we adopt an emerging concept, mobile edge computing, to reduce communication latencies by bringing end services to MECs located at network edges. However, the achieved low latency is at the expense of considerable operational cost from a large number of required MECs. To this end, optimal resource management approaches that can dynamically exploit the underlying infrastructures' network resources are required. In the rest of this chapter, we investigate the design of dynamic resource management for low latency applications.

4.1 Introduction

Over the last decade, advances in wireless access technologies (e.g., WiFi and LTE) have enabled an explosion of resource-hungry mobile applications, challenging current mobile devices' processing ability. In particular, mobile multimedia services with stringent latency requirements (in the order of hundreds of milliseconds [1]), such as AR, high-definition video streaming [118], gaming and face recognition, are computationally expensive for today's mobile devices; resulting in fast exhaustion of battery life and long processing delays [119]. Conventional cloud solutions [15], where users exploit preallocated service instances from data center-based clouds to process computationally expensive tasks, address the issue of computational resources, but suffer from long network latencies [5]. On the other hand, MEC (also known as *cloudlet* [5], *fog computing* [120], *Telco cloud* [121], *follow-me cloud* [122]) mitigates the long network latency issue by deploying dedicated micro-clouds along with service instances at network locations that are closer to users, e.g., APs, routers, etc.

However, since the micro-clouds are deployed at *fixed* locations and have limited physical resources (especially compared to data center-based clouds), they are deployed to large number of APs with MEC service instances in each micro-cloud [123]. This achieves low latency at the expense of significant operational costs due to break of DC consolidation [15, 119]. Limiting the number of micro-clouds can save operational costs, but faces challenges in dynamically supporting low latency services with limited resources at static network locations. For instance, current resource allocation techniques to deal with workload elasticity, such as auto-scaling [57, 124], could only scale up to the physical capacity limit of micro-clouds. Subsequently, if there is no micro-cloud in the vicinity of the overloaded one that can provide more computational resources for load balancing, users' tasks would accumulate, leading to the violation of the required service response time (e.g., time spent in network and edge clouds).

Recently, NFV was proposed to facilitate network function deployment for ISPs [117]. It decouples network functions from the underlying hardware and implements them as software in VMs hosted in commodity servers. The advent of NFV promotes the emerging concept of NFV-enabled MEC (e.g., [14, 125]) whereby services can be hosted at any network location that has virtualized resources, e.g., provided by commodity servers. Such NFV-enabled MEC model enables real-time instantiation (e.g., VM instantiation time for Unikernel [126] and

ClickOS [127] are in the order of tens of milliseconds) of MEC at new network locations to host edge services, and also allows MEC to scale up/down computational resources to accommodate user demand variations. As a result, the MEC can be dynamically instantiated at network locations that efficiently utilize ISPs' virtual network infrastructures and thereby maintaining low operational costs overtime. However, such flexibility in resource allocation faces challenges in:

- *Dynamically deriving the MEC service-hosting locations, amount of resources and the corresponding network paths* to mobile users such that the resulting network access latencies are within the network latency requirements and the ISPs' virtualized network resources are optimally utilized.
- *Determining the appropriate time instance* to perform dynamic resource allocation in order to avoid computation congestion at VMs due to peak load [128].
- *Performing resource allocation in a timely manner* such that the time spent in deriving a resource allocation decision does not affect low latency MEC services.

Unlike Chapter 3 where the offline physical resource allocation (i.e., network planning) was investigated for low latency networks, this chapter aims to achieve low latencies in a cost-efficient way through dynamic resource management of virtualized network resources. In particular, we take into account the flexibility afforded by NFV along with the abovementioned challenges, and study the problem of dynamic resource allocation in MEC, aiming at minimizing operational costs while satisfying users' low latency service response time requirements.

For the above problem, we propose a novel dynamic resource allocation framework for NFV-enabled MEC that consists of an online heuristic-based incremental allocation mechanism and a global resource reoptimization algorithm to address the trade-off between cost efficiency and low latency requirement. In particular, our online heuristic-based incremental allocation mechanism aims to efficiently allocate resources to tackle local MEC computation congestion due to (sudden) increase of workload in a timely manner. It consists of (1) an initial offline MEC resource allocation based on expected workload that achieves the desired service response time with the minimum required computational resources, (2) an auto-scaling and load balancing (ALB) mechanism that accommodates workload variations, (3) a *capacity violation detection* (CVD) mechanism that derives the projected time when ALB fails to cope with service elasticity and (4) a network latency constraint greedy (NLCG) algorithm of polynomial complexity to derive a new NFV-enabled node as MEC service-hosting node which supports the stringent latency requirement. Since our online allocation mechanism computes local MEC resource allocation, we also design a set cover partition approximation (SCPA) algorithm that operates in parallel with NLCG to *globally* reoptimize the locations and allocated resources while achieving a guaranteed operational cost. Given user demands, this cost is no more than $\ln(N)$ times of optimal MEC operational cost, where N is the largest number of APs that are served by a MEC service-hosting node among all instantiated MECs.

To demonstrate the effectiveness of our proposed framework, we carry out an extensive simulations with realistic three-layer cellular network setup [64]. We use real mobility traces from [129] to show the cost reduction brought by NFV-enabled flexible MEC instantiation compared to fixed-location MEC. Further, we conduct an in-depth cost efficiency impact factor analysis to give detailed insights into the design of online MEC resource allocation framework under various network topologies, latency requirements and server capacities.

The main contributions of this chapter are as follows.

1. We formulate and solve the dynamic MEC resource allocation problem as an ILP problem taking into account the flexibility in the determination of MEC locations enabled by NFV (see Section 4.2.2) and the trade-off between service response time and operational costs. To the best of our knowledge, this is the first study focusing on the dynamic MEC resource allocation taking into account the possibility of NFV-enabled MEC service instantiations.

2. We design a dynamic resource allocation framework consisting of a fast heuristic-based incremental allocation mechanism and a SCPA reoptimiza-

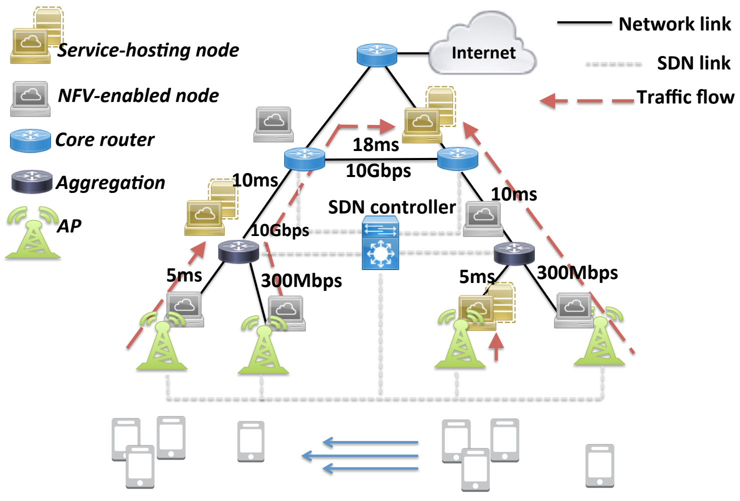


Figure 4.1: Hierarchical MEC system model.

tion algorithm for low-cost MEC resource allocation framework (see Section 4.3). Both NLCG and SCPA algorithms are general in nature and applicable to any online edge cloud systems (e.g., for different hosted services, edge cloud capacities and VM technologies). In addition, we mathematically prove that given user demands, our SCPA algorithm results in no more than $\ln(N)$ times of optimal MEC operational cost in polynomial time.

3. We demonstrate the effectiveness of our framework (see Section 4.4) through extensive simulations. We show that our framework achieves 33% cost reduction compared to fixed-location MEC overprovisioning solutions. Further, our in-depth impact factor analysis shows that SCPA achieves cost efficiency within 20% of the lower bound of the optimal solution, under different network size, services' latency requirements and MEC server capacities.

4.2 System Model and Problem Formulation

4.2.1 System Model

We consider a typical three-layer hierarchical wireless metropolitan area network [64] that consists of APs, aggregation nodes and metropolitan level mobile core network nodes (illustrated in Fig. 4.1). Each AP is connected to a single aggregation node which is connected to one mobile core node. Furthermore, the connectivity between mobile core nodes depends on the actual mobile core network's topology. For most of real-world topologies, a mobile core node has at least one network link towards other mobile core nodes (e.g., a topology example is shown in Fig. 4.2). We use $G = (V, E)$ to denote this network, where V is the set of network nodes and E is the set of links. Further, let B denote the set of APs, $b \in B$, which is a subset of network nodes ($B \subset V$). We consider that each network node is equipped with a commodity server [117], which has limited computational resources, k_v (e.g., CPU[1]) to host application service providers' services as software via VMs. Such support of MEC services with NFV-enabled nodes necessitates NFV commodity servers to be active (e.g., active servers are shown in Fig. 4.1 as service-hosting nodes) and hence, incurs operational costs (e.g., energy consumption) [15]. For the rest of paper, we consider MEC nodes to be any NFV-enabled network nodes on which MEC services can be hosted with allocated VMs.

Given the NFV-enabled MEC, mobile users upload raw files at discrete time, $t \in T$, through their associated APs to MEC nodes for processing rather than executing service instances locally in their mobile devices. The user requests from an AP are served by VMs at a single MEC node through the same path, $p_{bv} \in P_{bv}$, between AP b and node v (v is the selected node to host the required service)[2], where P is the set of paths between pair of nodes in V and $P_{bv} \subseteq P$. We use A_b^t to denote the total load incurred by mobile users at AP, b at time t, which results in bandwidth consumption, w_b^t, of flows departing from AP b. At the same time, user flows consume computational resources from MEC nodes, which depend on the AP-to-MEC assignment.

We consider stateless mobile services (e.g., AR, etc.) to be pre-installed as software into NFV-enabled nodes [1, 29]. That is, user requests can be seamlessly served by VMs at different MEC node without requiring service state migration since the services are stateless. In addition, NFV-enabled nodes that are not serving

[1]We only consider CPU as computational resources in this chapter.

[2]Multiple network paths between b and v could exist due to connectivity between mobile core nodes (see Fig. 4.2).

Table 4.1: Notations for dynamic resource allocation problem

Symbol	Notations
V, E, B	Set of NFV-enabled nodes, edges and APs
P,P_{bv}	Set of paths, set of paths between b and v
k_v	Resource capacity at node v
w_b	Bandwidth consumption at AP b
BW_e	Bandwidth capacity at network link e
$t \in T$	Discrete time slots
A_b^t	User computational resource demand from AP b at time t
D	Maximum network latency (hops) constraint
d_{bv}	Network hop distance between AP b and node v
N_b	The set of v that are located less than D network hops to b $N_b = \{v \mid d_{bv} \leq D\}$
$\mathscr{AP}_v$	The set of APs covered by network node v
$\mathscr{AP}_{v'v}$	The set of APs covered by network node v' and v
L_v	The excess workload from node v
$X_{p_{bv}}$	The path decision variable for $p_{bv} \in P_{bv}$
Y_v	The MEC node decision variable for v

as MEC nodes can instantiate VMs to support stateless MEC services in a timely manner. This is due to the latest advances in VM technology such as Unikernel [126] and ClickOS [127], whereby the VM instantiation time could be reduced to tens of milliseconds (e.g., 30ms [3]). We summarize the notations used in this paper in Table 4.1.

4.2.2 Operational Cost Minimization Problem

Given the abovementioned system model and the flexible instantiation of MEC nodes, we consider the MEC operational cost minimization problem for stateless low latency mobile services, whereby the network locations that host MEC services and the corresponding network paths can be dynamically controlled to efficiently utilize ISPs' resources. To better illustrate this scenario, an example is given in Fig. 4.2. We can see that two MEC nodes are instantiated among all NFV-enabled nodes together with its selected network paths at t_0. In contrast, only one MEC node is instantiated for operational cost minimization at t_1 in response to the decreased

[3]Unikernel, designed for edge computing environment, achieves 30ms by exploiting a shared memory channel to optimize the VM instantiation time.

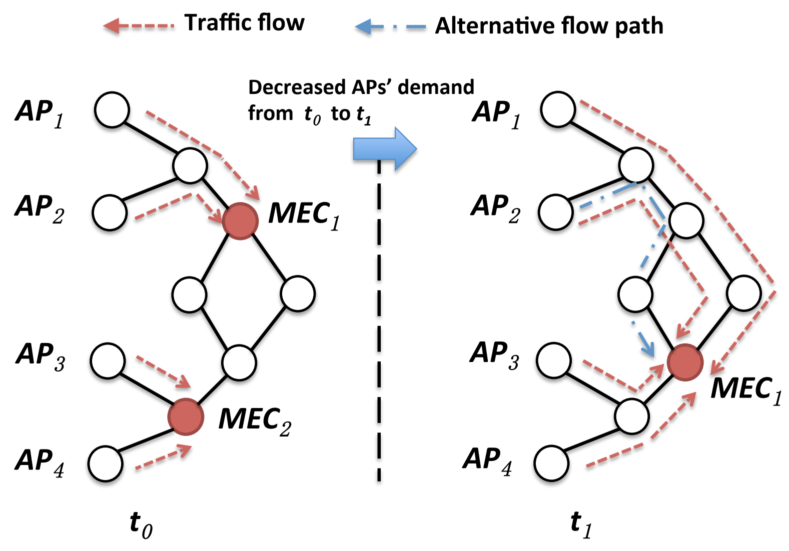


Figure 4.2: Example of MEC operational cost minimization problem.

demands from APs. Meanwhile, the network paths are accordingly changed at t_1.

In this chapter, we aim to concurrently answer four primary questions: given a time varying workload, resource-constrained distributed NFV-enabled network nodes and capacitated network links, (1) *where* and (2) *when* to allocate resources, (3) *how many* resources to be allocated among NFV-enabled nodes and (4) *which* network paths to use (e.g., between APs and MECs) such that the low latency requirements of mobile services are always satisfied while incurring the least operational cost. Without loss of generality, we assume in this work that all NFV-enabled commodity servers are identical (e.g., same specifications) and incur equal operational cost. Therefore, the operational cost minimization objective is equivalent to the minimization of number of active commodity servers (MEC node) [15].

We use ILP to formulate the problem with two binary decision variables, Y_v^t and $X_{p_{bv}}^t$, which represent respectively the location of MEC service (i.e., $Y_v^t = 1$ if at time t, v is chosen as the location of a MEC service and $Y_v^t = 0$ otherwise) and the path between b and v (i.e., $X_{p_{bv}}^t = 1$ if p_{bv} is chosen; $X_{p_{bv}}^t = 0$ otherwise). The objective function of the ILP is to minimize the number of selected MEC nodes, that is, the sum of $Y_v, v \in V$ at every discrete time instance[4], $t \in T$.

[4]Note that by fixing $T = \{t_0\}$, the problem is reduced to a static placement problem mentioned in Section I.

To satisfy the service latency requirement, we first decompose the request response time into the following:

1. *Network access time* – represents the time a MEC service request spent during network transmissions, which highly depends on the selection of network path, $X^t_{p_{bv}}$, between an AP and the selected MEC node. To model such delay, we assume that as long as the capacities of the constituent links in the selected network path are not violated by MEC flows, we can represent access delay as a function of network hops. Hence, in order to achieve a required network access time, both link capacity and the number of network hops that the request traverses need to be constrained.

2. *Service processing time* – refers to the time a VM uses to serve a request. We assume that as long as there is an available resource unit, and the request rate is lower than service rate, the processing delay is bounded and can be represented by a mean expected value that depends on the actual VM technology. To satisfy the processing time, we constrain the aggregated resource demands from APs that are served by MEC node at time t to be no more than its physical capacity limit. This ensures a fixed service time at all time by allocating a dedicated resource unit for each request.

The ILP problem is formulated as below:

$$\min \sum_{v \in V} Y_v^t, \forall t \in T, \tag{4.1}$$

Subject to

$$\sum_{b \in B} \sum_{v \in V} \sum_{p_{bv}(e) \in P_{bv}(e)} w_b^t X_{p_{bv}(e)}^t \leq BW_e, \forall e \in E, \forall t \in T, \tag{4.2}$$

$$\sum_{p_{bv} \in P_{bv}} \sum_{v \in N_b} X_{p_{bv}}^t = 1, \forall b \in B, \forall t \in T, \tag{4.3}$$

$$\sum_{p_{bv} \in P_{bv}} \sum_{b \in B} A_b^t X_{p_{bv}}^t - k_v Y_v^t \leq 0, \forall v \in V, \forall t \in T, \tag{4.4}$$

$$Y_v^t \in \{0,1\}, \forall v \in V, \forall t \in T, \tag{4.5}$$

$$X_{p_{bv}}^t \in \{0,1\}, \forall p_{bv} \in P_{bv}, \forall t \in T, \tag{4.6}$$

Constraint (2) guarantees that for all edges, the aggregated bandwidth consumption is less than the link capacity, BW_e, at every time instance, where $P_{bv}(e)$ denotes all paths between b and v that traverse edge, e; Constraint (3) guarantees that flows from the same b are assigned to the same MEC node v where v is selected from the set of network locations $N_b = \{v | d_{bv} \leq D\}$ that are within the network latency constraint denoted as D; Constraint (4) guarantees that the aggregated demands from APs at time t, $\sum_{p_{bv} \in P_{bv}} \sum_{b \in B} A_b^t X_{p_{bv}}^t$, served by the selected MEC v is no more than its physical capacity limit k_v and Constraints (5)-(6) limit the decision variables to be either 0 or 1.

Our problem stated above is NP-hard. A relaxed version of our problem (i.e., without the bandwidth capacity constraints (2)) can be obtained from the CSCP[5]. Since CSCP problem has been shown to be NP-hard [47], our problem is NP-hard too.

[5]In a capacitated set cover instance, we are given a universe X of n elements and a collection $\mathscr{S}$ of m subsets of X with elements having demand $d : X \mapsto \mathbb{R}^+$ and sets having supplies $s : \mathscr{S} \mapsto \mathbb{R}^+$, each subset has an associated cost; the objective is to pick the collection of sets $\mathscr{S}' \in \mathscr{S}$ of least total cost, such that each element $e \in X$ is contained in at least one set $S \in \mathscr{S}'$ while the supply of each set in $\mathscr{S}'$ is not violated [47].

4.3 Cost-efficient Dynamic Resource Allocation Framework

4.3.1 Overview

Our problem aims at deriving the optimal MEC locations, amount of resources and network paths to MECs in face of dynamic workloads to satisfy services' low latency requirements while minimizing the overall operational costs incurred within the time period, T. Offline solutions (e.g., overprovisioning) only solve the latency aspect of the problem while ignoring the possible high costs incurred due to inefficient resource utilization. Existing dynamic solutions are either based on local search or global optimization. The former derives the resource allocation in a timely manner by targeting specific network areas suffering from resource exhaustion which however often results in sub-optimal allocations. On the other hand, the latter takes demands across the whole network and is generally able to obtain better results at the cost of running time due to the large scale input from the entire network. Note that such long running time is not tolerable to online MEC as it would affect the performance of low latency services. To overcome the abovementioned issues suffered by most conventional approaches, we propose a novel dynamic optimization framework for NFV-enabled MEC that leverages both the local resource allocation and global re-allocation of resources to achieve a balanced trade-off between resource allocation optimality and algorithm's running time.

Fig. 4.3 presents the overview of our dynamic resource allocation framework.

1. *Heuristic-based incremental allocation mechanism* (see the right-side of Fig. 4.3) follows the local search principle and aims at deriving the minimum required resources for MEC in a timely manner in response to temporary workload increase. The idea is to first provision NFV-enabled MEC with the minimum (optimal) number of MECs to satisfy the average user demands. Then, it exploits conventional techniques for coping with (minor) service elasticity (i.e., ALB) to maintain the overall number of MECs at a relatively low level. At the same time, we detect the time point when these mitigation tools will

reach their limits (i.e., this implies that the existing MECs have been fully utilized) and cause the MEC system to violate the service response time requirement of the considered service(s). In such event, the allocation of a new MEC node (e.g., within the network latency constraints of APs that overloaded one of the existing MECs) will be chosen from the neighbouring network nodes of the overloaded MEC nodes (e.g., not searching the entire network), and activated in time before service quality degrades. By limiting the search scope to within the overloaded network area for the new MEC node, we significantly reduce the algorithm's running time and avoid service response time violations due to computation congestion at MECs. However, the heuristic-based incremental allocation solution has a major disadvantage due to the fact that it only incrementally adds MEC nodes to the existing MEC nodes that are previously allocated. As a result, the MEC resource allocation may gradually deviate from the optimum over time due to its lack of consideration for global workload variations.

2. *SCPA global reoptimization* (see the left-side of Fig. 4.3) aims to overcome the disadvantages of heuristic-based incremental allocation by adjusting the allocated resources at a coarse-grained time granularity to a near-optimal state. SCPA is periodically performed in a less frequent manner. It takes the resulting MEC nodes from the incremental solution and globally adjusts the resource allocation to maintain low MEC operational costs[6] within a bounded resulting operational cost.

Next, we elaborate on how these two approaches jointly solve the MEC operational cost minimization problem while always conforming to the latency constraint. Our framework follows the procedure below.

1. We derive the initial optimal static MEC placement (i.e., the number of MECs is minimized) in an offline fashion by solving the static version of the problem[7] at time t_0 using CPLEX [54].

[6]We do not consider migration costs as applications are stateless.

[7]For large scale problem, we solve the relaxed version of our problem, and derive the lower

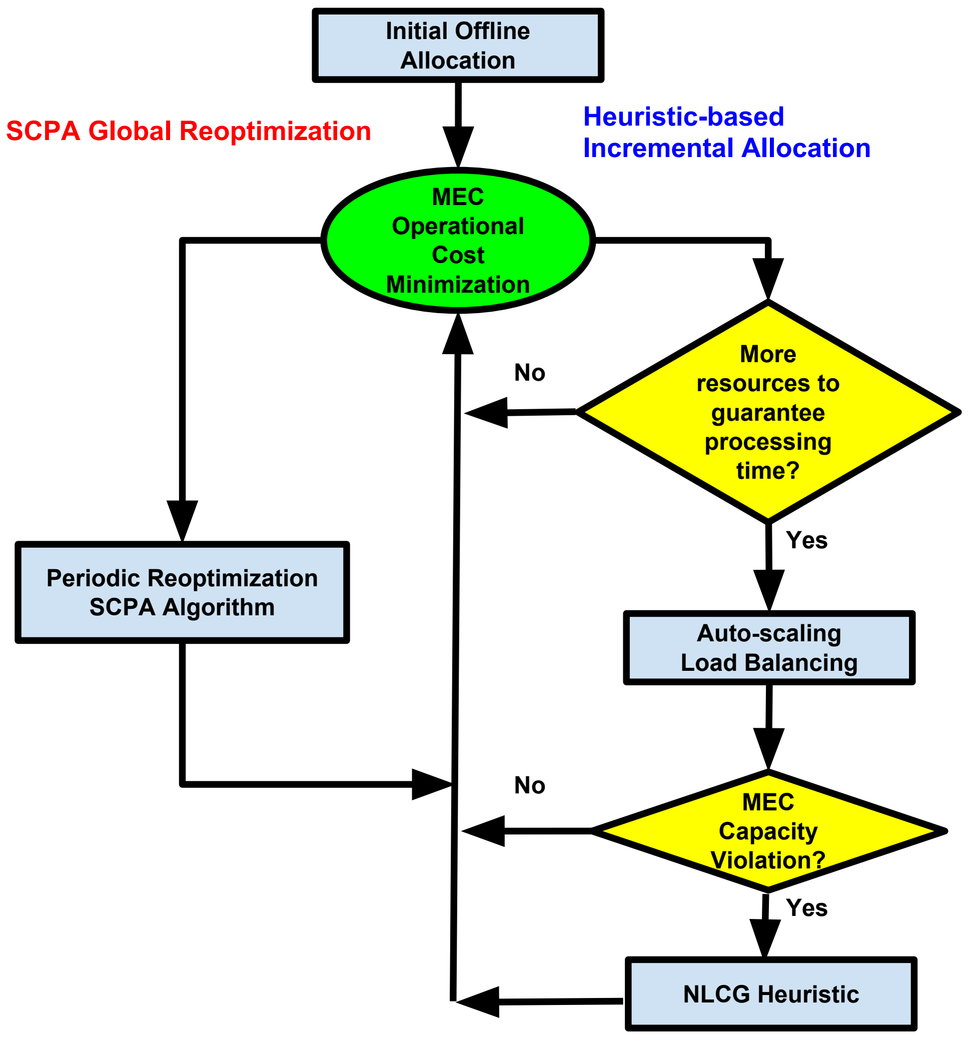

Initial Offline Allocation
SCPA Global Reoptimization
Heuristic-based Incremental Allocation
MEC Operational Cost Minimization
No
More resources to guarantee processing time?
Yes
Periodic Reoptimization SCPA Algorithm
Auto-scaling Load Balancing
No
MEC Capacity Violation?
Yes
NLCG Heuristic

Figure 4.3: Dynamic resource allocation framework overview.

2. We leverage conventional ALB mechanisms to cope with service elasticity based on the initial or most current placement and allocation such that the service processing time is guaranteed (i.e., no computation congestion at MECs) and the overall MEC number (e.g., operational cost) is kept low.

3. When the workload approaches the cloud capacity threshold, the system triggers the CVD mechanism based on the projected workload over a time window $\Delta t = t' - t$ where t' is the prediction time slot. Note that Δt will be selected according to the size of MEC network and the hosted mobile applications in MECs.

bound of optimal solution.

4. If it is detected that the ALB's limit will be reached within the coming time horizon, Δt, our NLCG algorithm is invoked to derive the desired new MEC node allocation based on the previous allocation solution. By appropriately deriving the NLCG start time, we minimize the added MECs in face of dynamic workloads.

5. A global reoptimization algorithm is performed periodically to adjust the MEC locations of the *entire* network, allocation of MEC nodes and the corresponding network paths such that given a certain user demands, the MEC operational cost is bounded.

4.3.2 Heuristic-Based Incremental Allocation Mechanism

In the following, we detail every component of our heuristic-based incremental allocation mechanism.

4.3.2.1 Static Offline Resource Allocation

We first derive the minimum required number of MEC nodes, its network locations, amount of allocated resources and AP-to-MEC network paths with CPLEX to support the low latency requirement given the average / expected user demands. We highlight that the offline resource allocation takes place at the network planning stage which does not impose any optimization execution time constraints. However, when the input size to CPLEX is extremely large (e.g., more than 300 network nodes), a relaxed version [8] of the MEC operational cost minimization problem is solved to get a feasible solution within polynomial time.

4.3.2.2 Auto-Scaling and Load Balancing (ALB)

Auto-scaling and load balancing are two current existing cloud computing elastic techniques to accommodate dynamic workload variations. We adopt a reactive auto-scaling solution that is triggered once a specific capacity threshold is reached. However, auto-scaling incurs additional VM reconfiguration delays which could affect service response time. This effect can be mitigated by setting a smaller auto-

[8]We relax the routing decision variable (i.e., from integer to linear programming).

Algorithm 4 Capacity Violation Detection (CVD)

```
Input: $G(V,E), B$, predicted workload $A^{t'}, v', k_{v'}$
Output: Future time $t'$ and extra load $L_{v'}$ or no NLCG
1: if current MEC nodes cannot accommodate $A^{t'}$ then
2:     Derive new AP-to-MEC assignments and resource
       allocation with $VALB$
3:     if $VALB$ cannot handle $A^{t'}$ then
4:         Derive $L_{v'}$ by $A^{t'}$, the new assignments and
5:         capacities of MEC nodes
6:         Trigger NLCG algorithm return $t', L_{v'}$
7:     else
8:         Perform $ALB$
9:     end if
10: end if
```

scaling threshold to invoke the auto-scaling mechanism in advance. Alternatively, proactive auto-scaling [73] can be applied to mitigate such auto-scaling overheads.

For load balancing, we adopt a proximity-aware solution [124] that considers both the residual capacity in MEC nodes and the topological proximity between MEC nodes and APs. Specifically, a flow from an AP to the overloaded MEC node will only be redirected when the newly chosen MEC node, v, is within the network latency cover, N_b, and the residual capacity is sufficient to accommodate the redirected load. By doing so, the network latency and MEC processing time are always bounded after load balancing.

4.3.2.3 Capacity Violation Detection (CVD) Mechanism

ALB have their limits, after which further increase in the request rate will incur increasing queuing delays at MEC nodes and lead to potential latency violations. The core idea of the CVD mechanism is to identify the time when such limitations will be reached so as to allow the system to pro-actively allocate new MEC node(s). Algorithm 4 presents the pseudocode of the CVD mechanism.

For CVD, we first assume that the workload can be reasonably predicted (e.g., perfect prediction). In practice, prediction algorithms predict workloads based on historical workload data. Algorithms such as generalized autoregressive conditional heteroscedasticity model [57] and various more [73] can be accommodated into CVD. We note that prediction techniques are not the main focus of this work. Given the current MEC node locations, resource utilization level and AP-to-MEC assignment, we predict over the time window Δt the aggregated workload $\sum\limits_{p_{bv'} \in P_{bv'}} \sum\limits_{b \in B} A_b^{t'} X_{p_{bv'}}^t$ at v' (i.e., v' is the MEC node that invokes the detection) and check if the predicted workload results in a capacity violation at v' (Line 1 in Algorithm 4). If the current state is predicted to be insufficient to accommodate the projected workload, we then estimate the future system state by virtually running ALB on the current system state with the projected workload.

The virtual ALB (VALB) aims to fully exploit computational resources provided by MEC nodes located in different network locations before triggering NLCG. It checks if load (e.g., offloading tasks from the same AP) from v' could be redirected to other MEC nodes while still conforming to the response time requirements of these flows. If virtual load balancing fails, virtual auto-scaling will be triggered to check if it can accommodate additional workloads by invoking auto-scaling. If this fails again, it means ALB will reach its limit within the projected time horizon and the overloaded MEC needs more computational resources to guarantee the service performance. Then, CVD records the excess load that cannot be served by v' as $L_{v'} = \sum\limits_{p_{bv'} \in P_{bv'}} \sum\limits_{b \in B} A_b^{t'} X_{p_{bv'}}^{t'} - k_{v'} Y_{v'}^t$ and triggers the online NLCG heuristic. It is worth mentioning that VALB is running as a real-time simulation where no actual ALB and any network configurations take place.

4.3.2.4 Network Latency Constraint Greedy Heuristic

The NLCG algorithm simultaneously determines the new placement of MEC node(s), the required resources and the corresponding routes. The idea of NLCG (Algorithm 5) is to search for a new MEC node located within the applications' network latency constraints that can accommodate the excess flow, $L_{v'}$, from the overloaded MEC node v' within the projected time. At the same time, the newly selected MEC node needs to satisfy as many flows (e.g., flows from APs served by other MEC nodes) as possible without violating network access delay to increase potential gain via load balancing to the new MEC node.

Specifically, NLCG first derives, for each network node other than existing MEC node $v \in V_s$, the number of APs covered by both the overloaded MEC node v' and v. To this end, NLCG finds the set of APs, denoted by $\mathscr{AP}_{v'} = \{b | d_{bv'} \leq D, b \in$

Algorithm 5 Network Latency Constraint Greedy (NLCG)

```
Input: $G(V,E), B$ represents APs, existing MEC nodes $V_s$, latency constraint $D$,
    overloaded MEC node $v'$, excess flow $L_{v'}$, predicted workload $A^{t'}$
Output: newly selected MEC node(s) and the corresponding routes
 1: New MEC node initialization $v_{bmax} \leftarrow \emptyset$
 2: Find the set of APs, $\mathscr{AP}_{v'}$, located in the distance cover of overloaded MEC
    node $v'$
 3: For each network node $v \in V \backslash V_s$, find the APs, $\mathscr{AP}_{v'v}$ , that are located both
    in the cover of $v$ and $v'$
 4: for all $b \in \mathscr{AP}_{v'}$ do
 5:     for all $v \in N_b$ and $v$ not in $V_s$ do
 6:         if $v$ can accommodate excess flow $L_{v'}$ and
    $|\mathscr{AP}_{v'v}| \geq |\mathscr{AP}_{v'v_{bmax}}|$ then
 7:             $v_{bmax} \leftarrow v$
 8:         end if
 9:     end for
10: end for
11: if no MEC found $v_{bmax} == \emptyset$ then
12:     $v_{bmax} \leftarrow argmax(|\mathscr{AP}_{v'v}|)$
13:     trigger NLCG again with newly derived excess flow
        $L_{v'} = L_{v'} - k_{v_{bmax}}$
14: end if
15: Find network routes for the newly allocated MEC node(s) $X_{t'} \leftarrow$
    $MinMaxFaireness(v_{bmax}, \mathscr{AP}_{v'v_{bmax}})$
16: Update $Y^{t'}$ with $V_s \leftarrow V_s \cup v_{bmax}$
17: return MEC node locations $Y^{t'}$ and routings $X^{t'}$
```

$B\}$, within the latency coverage of the overloaded MEC, v' (Line 2 in Algorithm 5). Next, it adds all APs that are located within the distance cover of both v' and v into $\mathscr{AP}_{v'v} = \{b|d_{bv} \leq D, d_{bv'} \leq D, b \in B\}$ (Line 3). Then, for each AP within the distance cover $b \in \mathscr{AP}_{v'}$ of overloaded MEC v', NLCG searches the potential MEC node v from the candidate set $N_b = \{v|d_{bv} \leq D, v \in V\}$, and greedily chooses the node $v_{b_{max}}$ that has the highest $\mathscr{AP}_{v'v}$ and can support excess load $L_{v'}$ (Line 4-10). If no viable $v_{b_{max}}$ can be found, NLCG assigns the v that has the largest $\mathscr{AP}_{v'v}$ as v_{bmax} (Line 11-12). This means that there is no single node location that can host all the excess flows $L_{v'}$ from v'. In this case, NLCG will be triggered again with a reduced $L_{v'} = L_{v'} - k_{v_{b_{max}}}$ to find the next location to add (Line 13). NLCG then directs flows in $\mathscr{AP}_{v'}$ previously served by v' to $v_{b_{max}}$ and solve the routing problem using min-max fairness [22] (Line 15).

Upon completion of NLCG, VM instantiation will start at NFV-enabled servers that have been selected to serve as MEC nodes. This instantiation process needs to

Algorithm 6 Set Cover Partition Approximation (SCPA)

```
Input: $G(V,E), B$ represents APs
Output: MEC nodes and the corresponding routes
 1: $V_s \leftarrow \emptyset$ where $V_s$ is the set of MEC nodes
 2: while $V_s$ is not a feasible solution do
 3:     Select $v \in V$ that maximizes the increase of newly
        covered APs in $V_s$
 4:     Store newly covered APs by $v$ into $\mathscr{AP}_v$
 5:     $V_s \leftarrow V_s \cup v$
 6: end while
 7: for all $v \in V_s$ do
 8:     $f_v \leftarrow G.fractionalMaxFlow(v, \mathscr{AP}_v)$
 9:     Construct subgraphs $G_v(V_v, E_v)$ with edges and nodes
        traversed by $f_v$
10:     $G_v.partition(\mathscr{AP}_v)$ [130] finds the unsplittable flows
        between APs in $\mathscr{AP}_v$ and $v$
11: end for
12: Superimpose paths found in each subgraph $G_v$
13: return MEC node locations and routings
```

accomplish before application workload $A^{t'}$ arrives so that application's response time will not be affected by VM instantiation. In other words, the overall time of VM instantiation and NLCG running time needs to be smaller than CVD's detection interval. In our framework, since CVD interval (e.g., on the order of minutes [57]) is not on the same order as VM instantiation time (e.g., on the order of tens of milliseconds [126, 127]), the abovementioned condition can be achieved if NLCG's running time is fast. We will evaluate NLCG's running time and heuristic's resulting application response time in Section 4.4.

4.3.3 Global Optimal Reoptimization Algorithm

To complement our incremental allocation mechanism, we devise the SCPA reoptimization algorithm (see Algorithm 6) with guaranteed performance bounds where an approximation ratio is derived to indicate how far the obtained solution is from the optimal solution. The SCPA algorithm first finds the locations and resources of MEC nodes by solving a CSCP with each MEC node being assigned a subset of demand nodes (e.g., APs) without considering the capacity constraint of each link in the network. Clearly, this solution does not represent a feasible solution to our original problem, as the network link capacity constraint and AP-to-MEC paths are not incorporated. To obtain a feasible solution, SCPA then applies a graph partition technique to find the routes between each AP and MEC node that are assigned such that the link capacity constraint is satisfied. Specifically, we decompose the original MEC operational cost minimization problem into a CSCP and a set of single-source unsplittable flow problem (SSUFP)[9]. The solution to the CSCP gives MEC node allocation and the corresponding AP assignment, while the solution to each SSUFP derives the specific path between each MEC node and its assigned AP.

MEC node selection: We first show how the MEC node allocation for delay-sensitive applications without bandwidth constraints is transformed into a CSCP problem. To this end, we consider each network node $v \in V$ as a set in the CSCP problem, and its computational capacity represents the supply of the set. An AP b denotes an element in the CSCP problem, and it can be covered by v if the network latency constraint is satisfied with $d_{bv} \leq D$. The number of requests at b denotes the demand of its corresponding element in the CSCP problem. Without loss of generality, we assume that the total demand of all APs can be fulfilled by the total resources available in the network. Then, the MEC node allocation without bandwidth constraints but with latency constraints becomes finding a capacitated set cover for the CSCP problem. Let V_s be such a feasible solution to the CSCP problem, which can be found by utilizing the algorithm due to [46]. Each network node in $v \in V_s$ is selected to serve as a MEC node, and the APs, $\mathscr{AP}_v$, that are within its range in terms of network latency, will be covered by the MEC node allocated at v. The procedures of finding each MEC node $v \in V_s$ is described in Algorithm 3 (Line 2-5), whereby the basic idea is to find a network node at each iteration that covers the most of APs until all APs are assigned to one of the selected network node in V_s

.

Network path selection: Next, we proceed to find the paths between each of the selected MEC node $v \in V_s$ and its covered APs, $\mathscr{AP}_v$, where the bandwidth resource constraint of each link in G is taken into account. We first get a frac-

[9]In a single-source unsplittable flow instance (SSUFP), we are given a network $G = (V, E)$, a source vertex s, a set of k commodities with sinks $t_1, ..., t_k$ and the associated real-valued demands $\rho_1, ..., \rho_k$. The objective is to route the demand ρ_i of each commodity i along a single $s - t_i$ flow path so that the total flow routed across any edge e is bounded by the edge capacity BW_e.

tional maximum flow f_v[10] for each MEC node $v \in V_s$ and its assigned APs in $\mathscr{A}\mathscr{P}_v$ (Line 8). Based on f_v, we construct $|V_s|$ subgraphs $G_v(V_v, E_v)$ by including v, its assigned $\mathscr{A}\mathscr{P}_v$, all other intermediate network nodes (V_v) that connect v and its $\mathscr{A}\mathscr{P}_v$, and the links (E_v) traversed by f_v (Line 9). We then find SSUFP in the constructed subgraph for each selected network node $v \in V_s$, by using the algorithm *PARTITION* described [130] (Line 10). The basic idea of algorithm *PARTITION* is to further partition each subgraph into ε subgraphs by including APs that have demands in the same demand interval and the corresponding fractional paths from f_v. Then, in order to find a feasible unsplittable path for all APs in each new subgraph, *PARTITION* updates edge capacities in each newly obtained subgraph by rounding up APs' demand to the upper bound of its demand interval (i.e., this leads to the increase of edge capacity in subgraphs). Next, *PARTITION* iteratively applies augmenting path algorithm to find a feasible (e.g., conforms to augmented link capacities) unsplittable path for each AP. Finally, we superimpose unsplittable flows' solutions of each subgraph G_v to obtain the complete network paths (Line 12) for all APs. However, *PARTITION* violates at most $(4+\varepsilon)$ relative edge capacity for any $\varepsilon > 0$, where $n\frac{1}{2}^{\xi-1} \leqslant \varepsilon$ and ξ represents the number of partition intervals in algorithm *PARTITION*.

4.3.4 Algorithm Analysis

In this section, we derive the performance bounds of our SCPA global reoptimization algorithm detailed in Section 4.3.3. For this purpose, we will first re-state the following Theorems 4 and 5 given in [46] and [130] respectively.

Theorem 1. *[46]: Given a CSCP, there exists a greedy algorithm that finds a* $\ln(N)$ *approximation solution within running time of $O(|V|)$, where N gives the largest number of APs served by a MEC node in V_s.*

Theorem 2. *[130]: Given an UFP, algorithm PARTITION finds a $(4+\varepsilon)$ approximation for relative congestion for any $\varepsilon > 0$. The running time of the algorithm is*

[10]Note that maximum flow is a common problem where many different solutions can be applied (e.g., augmenting path algorithms [131]).

$O(T_1(|V|,|E|)+|V||E|+|E|\varepsilon)$, where $T_1(|V|,|E|)$ is the time to solve a fractional maximum flow problem.

Using the above, we can state the following theorem for our global reoptimization algorithm:

Theorem 3. *Given a NFV-enabled network environment, $G(V,E)$, where network node $v \in V$ has virtual computational resources k_v, network edge $e \in E$ has bandwidth BW_e, and APs $b \in B, B \subseteq V$ has user demands A_b, there is a fast approximation algorithm for the delay-guaranteed cost minimization problem that delivers a feasible solution with a cost no more than $\ln(N)$ times of the optimal cost in $O(|V|+|V_s|(T_1(|V|,|E|)+|V||E|+|E|\varepsilon))$ time, where N gives the largest number of APs served by a MEC node in V_s, $|V_s|$ gives the number of resulting MEC nodes and $T_1(|V|,|E|)$ is the time to solve a fractional maximum flow problem.*

Proof. We first show that the approximation ratio of our proposed SCPA algorithm is $\ln(N)$ times the optimal solution. Let C^* and C'^* be the optimal solutions to our problem with and without capacity constraints of network links.

The approximation solution to CSCP (Theorem 1) gives the lower bound of our original problem, i.e., $C'^* \leq C^*$. Specifically, in the aforementioned SCPA algorithm, the first step is to find the MEC node locations and the assignment of APs to the selected MEC nodes, which are given by solving the CSCP problem. Such node locations determine the resulting cost of both CSCP and our original problem defined in Section 4.2.2. However, CSCP does not answer through which paths the APs and MEC nodes are connected and the network bandwidth capacity constraints are ignored, which is a special case of our original problem. Hence, the solution to CSCP is the lower bound to the original problem.

Denote by C' and C the solutions of the first (node selection) and second (path selection) steps of the proposed SCPA algorithm. Clearly, we have $C' = C$, because in the second step no network nodes are included or removed. We thus have

$$\begin{aligned} C &= C' \\ &\leq C'^* \cdot \ln(N) \text{ , since Theorem 4} \\ &\leq C^* \cdot \ln(N). \end{aligned}$$

This means that the approximation ratio of the proposed algorithm is $\ln(N)$.

We then show that feasible unsplittable paths between MEC nodes and the APs assigned to each MEC node can be found in polynomial time and the resulting edge congestion is no more than $(4+\varepsilon)|V_s|$ times edge capacity.

The idea of showing the bound of edge congestion is by considering the worst-case where the edge that has the maximum flow in a subgraph G_v overlaps with all edges from other subgraphs that also have the maximum flow. This situation could occur as we partition the original graph into $|V_s|$ subgraphs after we solve CSCP, and an edge from the original graph G can be shared by many subgraphs. According to Theorem 5, the relative edge congestion is at most $(4+\varepsilon)$ in a subgraph G_v. Hence, the worst-case relative edge congestion in the original graph is at most $(4+\varepsilon)|V_s|$ since an edge in G_v can overlap with at most $|V_s|$ edges when it is superimposed with other edges.

We have now shown that there is a set of unsplittable flows for each subgraph G_v obtained from the solution to CSCP and each edge has a congestion no more than $(4+\varepsilon)|V_s|$. However, the edge congestion could violate the bandwidth constraint (2). This can be solved by setting the subgraph edge capacity by $\frac{BW_e}{(4+\varepsilon)|V_s|}$. Then, the edge capacity at all edges can be satisfied. Thus, the solution of the proposed SCPA algorithm satisfies all constraints and there is a feasible solution of paths for the lower bound (e.g., CSCP) of the original problem. This means that the approximation ratio of CSCP is the approximation ratio of the original problem.

Finally, we derive the running time of the proposed SCPA algorithm based on the running time from [130], where they showed solving a SSUFP requires a running time in $O(T_1(|V|,|E|)+|V||E|+|E|\varepsilon)$. More specifically, since our problem consists of solving a CSCP and $|V_s|$ SSUFP, we derive the running time by adding the running time of solving each subproblem. Therefore, the running time in our problem is $O(|V| + |V_s|(T_1(|V|,|E|) + |V||E| + \varepsilon|E|))$. $\square$

4.4 Performance Evaluation

In this section, we evaluate the efficiency of our proposed framework in terms of service response time (i.e., round-trip time and processing delay at VMs) and cost efficiency under different MEC settings (e.g., network size, application latency requirement and server capacity). We first show in Section 4.4.1 that our dynamic resource allocation framework achieves the low latency requirement of the application while resulting in lower operational costs compared to existing approaches. We then focus on the performance analysis of the SCPA reoptimization algorithm in Section 4.4.2. We compare SCPA's results against optimal and heuristic-based incremental allocation and show how close our SCPA algorithm can drive MEC systems back to the optimal state.

We clarify the schemes that will be compared against as follows:

1. *Overprovisioning* – We first solve the MEC placement and allocation at the peak workload with CPLEX in an offline manner and then, for each chosen location, we overprovision VMs with the maximum possible physical capacity to serve user requests, i.e., ALB and new MEC instantiation are never needed in this case.

2. *ALB* – We implement the initial solution from the static allocation problem at $t = 0$. The network performs ALB on the initial MEC locations (fixed locations) when needed.

3. *Heuristic* – Our proposed heuristic-based incremental allocation including NLCG algorithm, ALB and CVD (see the right-side of Fig. 4.3).

4. *Heuristic+Reoptimization* – Our proposed dynamic framework in full, combining heuristic-based incremental allocation and periodic SCPA global reoptimization that performs every 30 minutes.

4.4.1 Service Latency and Operational Costs

We use packet-level simulations to examine detailed MEC service latencies and operational costs. To this end, we create a realistic online NFV-enabled MEC simulation environment with OMNeT++ [132] complemented with an OpenFlow extension module provided by [133]. We implemented our dynamic resource allocation framework that operates as part of the centralized software-defined networking (SDN) controller. The controller connects to each network node through a dedicated network link (see Fig. 4.1), and dynamically carries out network configuration during MEC node instantiations.

We create a three-layer metropolitan wireless network shown in Fig. 4.1, consisting of APs, aggregation nodes and mobile core network nodes. In this network, the APs are deployed over an area of $46km^2$ where the deployment density is 0.65 APs per km^2. We further consider 1,800 mobile users moving following the mobility traces of a fleet of taxis operating in San Francisco [129]. Accordingly, we set up 30 APs, 5 aggregation nodes and 5 core network nodes (e.g, set according to part of Paris' core network model [134]) for the considered number of users and area where each network node is equipped with a cluster of commodity servers. In terms of server size, we follow [135] such that each network node has 21 servers and each server has 2.1GHz CPU of 18 cores. Moreover, we consider an AR application [29] where users upload street views captured by their mobile devices for annotations (e.g., building name, available parking places, etc.) computed by MEC. Such application requires a service response time of 480ms [1] and generates upload frames of size 0.5MB at 0.3FPS [29] which requires 230ms for a VM of 600MHz CPU to process [29]. For simplicity, we assume homogeneous frame size and upload rate for all users. In terms of network latency constraint, we set a maximum of 4 network hops[11] from AP to MEC node [136].

Given the aforementioned setup, we first derive the initial MEC node locations, resources needed and the corresponding network paths by CPLEX solver in an offline manner. Two MEC nodes are selected among all NFV-enabled nodes (the

[11] According to [136], when maximum number of network hops are no more than 4, MEC always outperforms DC-based cloud in terms of latency.

Table 4.2: Performance comparison with realistic topology.

	Latency Requirement	Maximum Latency	Number of MEC nodes (start)⟶(end)	Cost saving(%)
Overprovision	Succeed	480ms	3⟶3	0%
ALB	Fail	132s	2⟶2	42.6%
Heuristic	Succeed	480ms	2⟶3	33.6%
Heuristic+ Reoptimization	Succeed	480ms	2⟶3	33.6%

"Number of MEC nodes, (start)⟶(end)" column in Table 4.2 shows this number). Then, we execute our simulations for a duration of 1 hour from the abovementioned initial state, during which we gradually increase the AR application workload from 0.3FPS to the peak workload at 3.0FPS in steps of 0.1FPS every 400s. We set a threshold-based VM auto-scaling mechanism for our packet-level simulation. Whenever VM load reaches a threshold of 80%, auto-scaling mechanism is triggered with a VM instantiation time of 100ms, This is set according to a realistic NFV commodity servers' instantiation time following [126]. In addition, we set the workload prediction time window, $\Delta t = 400s$ [57] for the NLCG algorithm, and consider a 100% prediction accuracy. This assumption has been largely adopted in the design of online resource allocation algorithms [15, 64, 137]. On the other hand, an inaccurate workload prediction would result in overprovisioning or underprovisioning of MEC resources in practice, which leads to poor cost efficiency and long processing delay respectively. Many existing work such as [138] have studied the impact of prediction inaccuracy and the compensation techniques (e.g., [138] proposed a method to minimize the impact of prediction inaccuracy, in which they minimized the underprovisioning-caused latency violations less than 2% of all requests). Therefore, our evaluation focuses on the proposed algorithms.

Now, we compare our solution against existing solutions in terms of service latency and operational costs. Table 4.2 shows our results with respect to satisfaction of the response time requirement, number of resulting MEC nodes (i.e., operational costs) at the start and end of the simulation and the cost savings over time in comparison to the *Overprovision* scheme. From the table, we can see that only the costly *Overprovision* and our solutions (*Heuristic* and *Heuristic+Reoptimization*), manage to satisfy the delay requirement of the considered AR application. In addition, *ALB* results in the lowest number of MEC nodes at the end of the simulations, but it comes with delay penalties due to computation congestion at the two initial MEC nodes. Our solutions have all increased the resulting number of MEC nodes by 1 in response to the increased workload. When we compare the costs over time against *Overprovision*, *ALB* achieves a saving of 42.6%. In contrast, our *Heuristic* and *Heuristic+Reoptimization* lead to a more modest saving (i.e., 33.6% in both cases), but achieves the latency requirement by increasing the overall computational resources through the new allocation of MEC nodes. Such saving is achieved by minimizing the number of required MEC node instantiations whereby the CVD mechanism derives the time instance when the resources of MECs will be fully utilized and cannot accommodate more workloads. However, due to the packet-level simulator's limitation, only a small topology is evaluated, whereby the performance improvement of *Heuristic+Reoptimization* cannot be revealed (e.g., identical results of cost saving in Table 4.2).

In addition, we observe from the cumulative distribution function (CDF) of response time in Fig. 4.4 that the resulting response time of our solutions overlap with that of *Overprovision*. This further shows the seamless transition to the new system state, and *Heuristic* approach is fast enough to get VMs ready before workload arrives. On the other hand, *ALB* fails to conform to the latency requirement with 20% (see Fig. 4.4) of the overall requests exceed the latency threshold (maximum latency at 132s) due to insufficient physical capacity in the fixed limited number of MECs.

Our detailed packet-level simulator allows us to track and examine each and every individual request and response packet in the system. The tradeoff to this is the scalability of the simulator which constrained us to smaller scale simulations. To more comprehensively evaluate our solution, we further evaluate our framework, specifically on the benefits brought by global reoptimization algorithm, SCPA, with larger network topologies in the next section. Also, we thoroughly investigate the

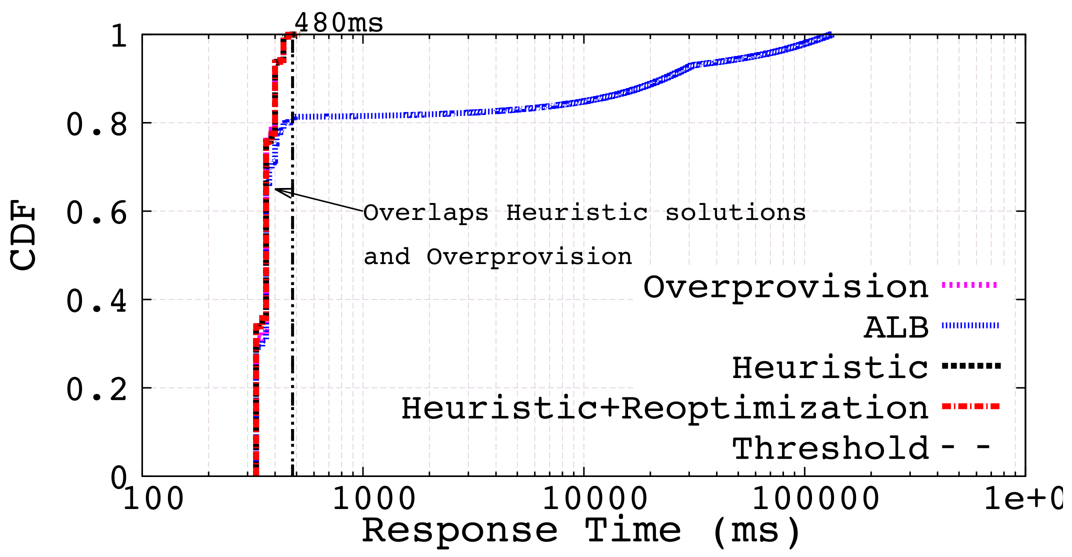


Figure 4.4: Response time.

impact of different network sizes, network hop constraints and MEC service-hosting servers' sizes to our solution.

4.4.2 System Cost Optimality

We proceed to evaluate the improvements provided by SCPA via flow-level simulations with large network topologies, and investigate by how much SCPA can drive the NFV-enabled MEC back to the optimal state. To this end, we compare the resulting MEC operational cost of *Heuristic+Reoptimization* against *Heuristic* and lower bound of optimal solution[12] denoted by OPT_{LB} under different network sizes, latency requirements and physical capacities of NFV-enabled servers. Furthermore, in order to more intuitively present SCPA reoptimization's optimality difference to OPT_{LB} and take into account MECs' resource utilization level, we introduce two

[12]Such OPT_{LB} is solved by relaxing both the edge capacity constraint and the routing decision variable X_p (i.e., from integer to linear programming). Note that this is a conservative estimation of the optimal solution, which is smaller than the optimal value. In addition, due to the complexity in deriving the OPT_{LB} solutions for large size networks (e.g., larger than 300 nodes), we stop the CPLEX solver when the optimality gap reaches 5% to avoid long execution time.

metrics: *cost efficiency* and *cost efficiency gap*. The cost efficiency, C_{eff}, quantifies the number of mobile users per MEC node who achieve the required service response time.

$$\text{Cost efficiency, } C_{eff} = \frac{Nb_{users}}{|V_s|} \tag{4.7}$$

where Nb_{users} is the total number of users who receive their services within the services' latency requirements.

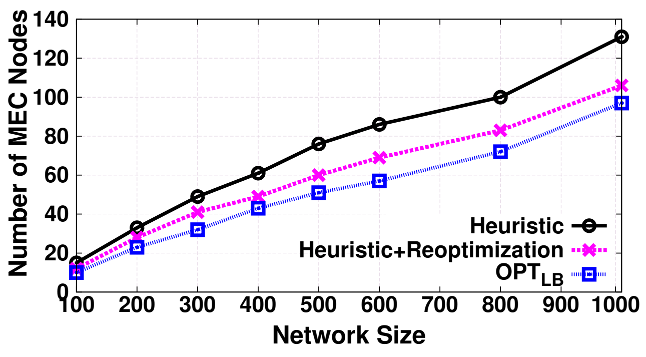
Cost efficiency gap shows how close the resulting cost efficiency of our solutions (i.e., *Heuristic+Reoptimization* and *Heuristic*) is to the OPT_{LB}, that is, the smaller this gap is, the more cost-efficient the solution is. More specifically, this metric is derived as the normalized difference between cost efficiency of our solutions and that of OPT_{LB}.

$$\text{Cost efficiency gap, } Gap_{eff} = \left| \frac{C_{eff}^{OPT_{LB}} - C_{eff}}{C_{eff}^{OPT_{LB}}} \right| \tag{4.8}$$

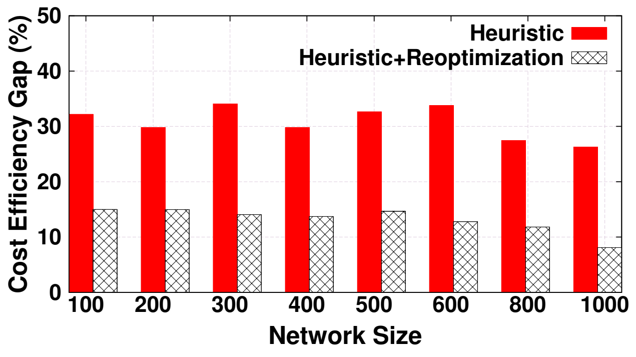
where $C_{eff}^{OPT_{LB}}$ denotes the cost efficiency of OPT_{LB}.

4.4.2.1 Impact of Network Size

We adopt GT-ITM [139] to generate synthetic network topologies where the probability of having an edge between two nodes is 0.2 with edge capacities uniformly distributed between 300Mbps and 10Gbps. Other setup / parameters related to the application, workload and server capacity remain the same as previously described (see Section 4.4.1). We plot in Fig. 4.5(a) the average number of MEC nodes in function of different network sizes ranging from 100 nodes to 1000 nodes for *Heuristic*, *Heuristic+Reoptimization* and OPT_{LB}. It must be stressed that the average number of MEC nodes at each network size (e.g., 100 to 1000 nodes) is the average number of MEC nodes of 4 simulations with different service latency requirements (e.g., maximum number of hops from 1 to 4 hops). By doing so, the



(a) Average costs for each network size over different latency requirements.



(b) Cost efficiency gap to OPT_{LB}.

Figure 4.5: Impact of network sizes to costs.

impact of a specific latency requirement to the MEC node number is reduced, and hence Fig. 4.5(a) can reflect the impact of network sizes to MEC node number in a more accurate way.

From Fig. 4.5(a), we see that the *Heuristic+Reoptimization* solution achieves lower operational costs (i.e., lower number of resulting MEC nodes) for all network sizes compared to *Heuristic*. The resulting MEC operational cost of our *Heuristic+Reoptimization* also closely follows that of OPT_{LB}. The relative poorer performance achieved by *Heuristic* is due to its local search nature where the search of a new MEC node is triggered by overloaded existing MEC nodes and carried out in the vicinity of these affected nodes. As a result, the optimal MEC location that may benefit the maximum number of users could potentially be omitted during

Heuristic's search process, leading to a relatively lower cost efficiency. In contrast, *Heuristic+Reoptimization* utilizes resources more efficiently by searching the optimal MEC locations over the entire network.

We show in Fig. 4.5(b) the cost efficiency gap to OPT_{LB} for *Heuristic+Reoptimization* and *Heuristic*. We see that *Heuristic*'s cost efficiency gap to OPT_{LB} is always above 25%, whereas *Heuristic+Reoptimization* can improve nearly 20% of *Heuristic*'s cost efficiency due to the global search. In addition, *Heuristic+Reoptimization* consistently achieves an efficiency gap below 15% for any network sizes (see Fig. 4.5(b)). In particular, we observe that *Heuristic+Reoptimization*'s efficiency gap does not increase with network size, which justifies the theoretical performance bound $\ln(N)$, whereby N represents the largest number of APs served by a MEC node, which is independent to the size of network.

4.4.2.2 Impact of Latency Requirements

The latency requirement can be interpreted as the maximum tolerable number of network hops between APs and MEC nodes. It directly affects the number of APs that a NFV network node can cover (i.e., serving the APs without violating latency requirements). This, in turn, affects the required MEC nodes to cover all APs in the proposed algorithms. To show the impact of this factor, we vary the maximum tolerable number of network hops from 1 to 4, which reflects latency requirements of different nature such as extremely strict network latencies (e.g., 10ms network delay) to loose latencies (e.g., 150ms network latency). We show, with Fig. 4.6, both *Heuristic* and *Heuristic+Reoptimization*'s cost efficiency gap ratio for each of considered latency requirement. Note that the cost efficiency gap at each latency requirement in Fig. 4.6 is the average of that of all network sizes (e.g., 100 to 1000).

We see from Fig. 4.6 that *Heurisitc+Reoptimization* still outperforms *Heuristic* for each latency requirement, and it always achieves an efficiency gap below 20%. In particular, when the maximum network hop is set to zero, both *Heuristic* and *Heurisitc+Reoptimization* achieve an optimal operational cost where the efficiency gap equals to zero. This is due to the fact that the extreme low latency constraint (e.g., 0 hop) restricts all APs to be served as MEC nodes, which makes

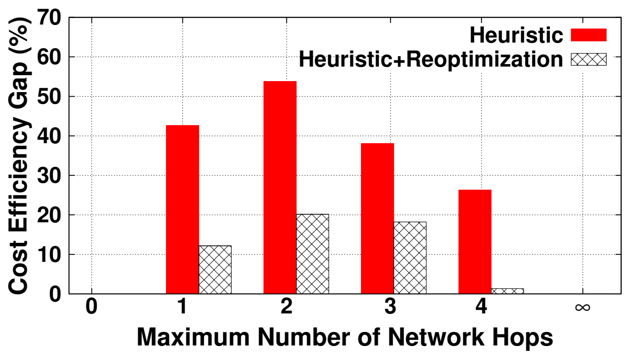


Figure 4.6: Cost efficiency gap to OPT_{LB}.

the resulting number of MEC nodes identical for any MEC allocation algorithms that conforms to the network latency constraint. Similarly, when we look at the other extreme case where the latency constraint is extremely loose (see ∞ in Fig. 4.6) and the physical NFV servers have infinite capacity, only one MEC node is required in *Heuristic*, *Heurisitc+Reoptimization* and OPT_{LB} (e.g., this leads to 0% cost efficiency gap). From the above two cases, we observe that the selection of MEC resource allocation algorithm does not play a critical role in the resulting MEC operational cost when latency is either extremely loose or strict. However, when the latency requirement is between the two extremes cases, it significantly affects the cost efficiency. For instance, when the latency requirement is set to 1, 2, 3 and 4 network hops, we observe from Fig. 4.6 that the cost efficiency gap of both *Heuristic* and *Heurisitc+Reoptimization* first increases and then decreases as the maximum tolerable network hops increase. The increase of cost efficiency gap at network hop 1 and 2 compared to 0 hop is due to the enlarged search space for MEC nodes in *Heuristic* and *Heurisitc+Reoptimization*. Such search space enlargement increases the chance of selecting less optimal MEC nodes where MECs' resource utilization is poorer compared to MECs derived by OPT_{LB}. On the other hand, the decrease of efficiency gap at 3 and 4 network hops is the consequence of improved MEC utilization compared to cases with 1 and 2 network hops. Specifically, due to the relaxed latency constraint, a MEC node can serve a larger number of users without violating the network latency requirement, and hence achieve a better resource utilization compared to strict latency requirements. In particular, the relaxed latency constraint at 4 network hops results in a situation where the number of served users in each MEC reaches servers' physical capacity limits, that is, the resource utilization at each derived MEC is almost 100%. Knowing that an optimal MEC resource allocation achieves the least number of MEC also by fully utilizing MECs' resources. Therefore, the fully utilized MEC nodes at 4 network hops achieve a very close cost to OPT_{LB}. Similarly, when the latency becomes even less strict (e.g., ∞), the allocated resources at MECs will reach the servers' physical capacity limits and result in the same operational cost as OPT_{LB} (see ∞ in Fig. 4.6). Clearly, there is an inter-correlation between applications' latency requirement and the server capacity, which we elaborate in the next subsection.

4.4.2.3 Impact of Physical Capacities

Next, we evaluate the impact of servers' physical capacities to *Heurisitc+Reoptimization*'s MEC costs. To this end, we consider three NFV-enabled servers sizes, namely, *FULL* (i.e., the considered server size (see Section 4.4.1)), *HALF* (i.e., half of *FULL* size), *DOUBLE* (i.e., two times the *FULL* size) [64, 135]. Furthermore, servers of different sizes result in different energy consumption, which can be estimated based on the server size and resource utilization [15]. We take a simplistic assumption in our evaluation whereby the energy consumption is proportional to the server size. That is, we consider *HALF* size servers consume half of *FULL* size servers' energy and correspondingly, *DOUBLE* size servers consume double the amount of energy of *FULL* size servers. We plot in Fig. 4.7 the average energy cost incurred by *Heurisitc+Reoptimization* under different network latency requirements for each of the abovementioned server sizes. We observe that simulations with *DOUBLE* server size result in higher costs than *HALF* and *FULL* server when network latency requirement is extremely low (e.g., 1 network hop). This is due to the inter-correlation between the two impact factors: latency requirement and server size. More specifically, when network latency requirement is extremely low, the latency requirement impact factor dominates the MEC node searching process leading to almost the same number and placement of resulting MEC nodes for

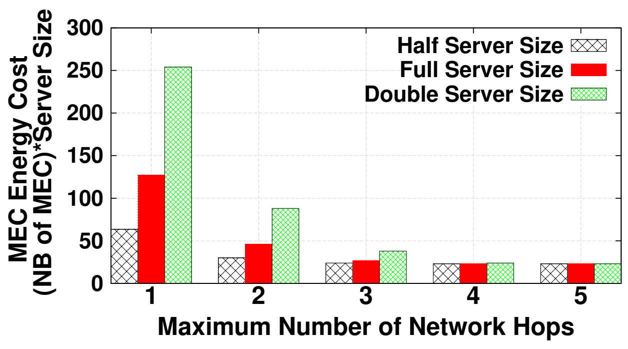


Figure 4.7: *Heuristic+Reoptimization*'s average cost for each latency constraint over different network sizes.

HALF, *FULL* and *DOUBLE* size servers. However, the per MEC energy consumption of *DOUBLE* size server is significantly more than that of *HALF* and *FULL* size which results in the overall higher costs (see Fig. 4.7). In contrast, when the latency requirement becomes less strict (e.g., network hops 3 and 4), *DOUBLE* size servers' energy cost decreases drastically as a consequence of decreased number of required servers and better resource utilization compared to that of strict latency requirements (i.e., each server supports a larger number of users within its network latency constraint). At the same time, we see from Fig. 4.7 that the resulting cost of the 3 server sizes converges to the same level after 3 network hops. For each server size, more users are served per MEC after the relaxation of latency requirements, and hence all MECs are almost fully utilized. As a consequence, the overall number of MEC nodes with full-size servers is half of that of half-size case and double of double-size case. Given the simplified energy cost assumption, our dynamic resource allocation framework results in the same level of energy consumption for each server size when latency requirement is loose.

Given the above observations, we see that the performance of dynamic resource allocation framework is independent of the network size. In particular, *Heurisitc+Reoptimization* can always improve *Heuristic*'s resulting operational cost except when the latency requirement is extremely low (e.g., 0 hops) or extremely high (e.g., ∞ hops). Also, the observations from the impact factor analysis

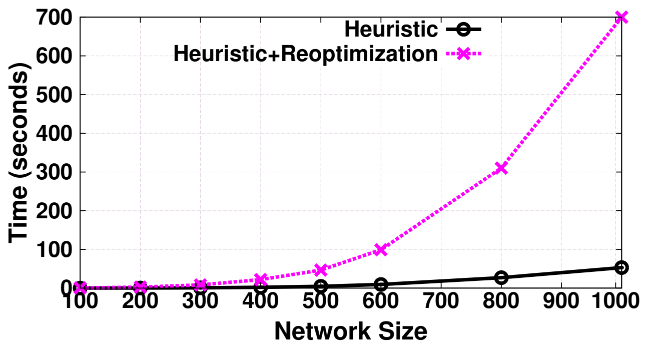


Figure 4.8: Algorithm running time comparison.

of latency and server capacity provide insights on the server size selection in the NFV-enable MEC cost minimization problem. We conclude that for extreme low latency applications (e.g., under 10ms), deployment of smaller servers are more desirable in order to achieve lower MEC cost through dynamic resource allocation. However, when the latency requirement is loose, the server size does not have strong influence on the MEC operational costs.

4.4.2.4 Algorithm Running Time

Last, we show in Fig. 4.8 the average running time of NLCG heuristic and SCPA reoptimization for each network size whereby the average running time is derived over different latency requirements. As Fig. 4.8 shows, the SCPA takes more time to execute than NLCG heuristic, but achieves a cost efficiency within 20% of OPT_{LB}'s cost efficiency (see Fig. 4.6). In addition, we observe that when network size is larger than 500 nodes, SCPA running time increases drastically due to the increased complexity in finding unsplittable flows. However, it must be stressed that conventional metropolitan-level wireless networks have network size smaller than 700 nodes [64, 76], and even the maximum execution time (e.g., 200s) for 700 nodes does not affect the desired latency requirements in the considered online NFV-enabled MEC (e.g., SCPA is performed less frequently than incremental MEC allocation in dynamic resource allocation framework). On the other hand, the NLCG's running time is below 50s in the worst case (e.g., network size 1000), which does not affect the latency requirements (i.e., the sum of VM instantiation time and NLCG's running time is always smaller than CVD detection interval).

4.5 Conclusion

Having resolved the cost-efficient low latency network planning problem in Chapter 3, this chapter addresses the dynamic resource management for cost-efficient low latency networks. That is, how to fully and dynamically exploit different network resources from the physical underlying network. Specifically, we adopt NFV-enhanced MECs to achieve both cost efficiency and low latencies by dynamically instantiating network resources at the optimal network locations. As such, the minimum amount of required network resources are allocated to achieve the required low latency, so that the operational cost of low latency networks can be minimized. For this, we formulate an optimization problem for allocating end services at any resource-constrained NFV-enabled MEC nodes. We demonstrate the effectiveness of our dynamic resource allocation framework in NFV-enabled MEC through both packet-level and flow-level simulations. Our results show that only our proposal always ensures that end services respond to user requests on time, while achieving up to 33% operational cost reduction in comparison to the current practices. Meanwhile, our proposal achieves a near-optimal MEC operational cost whereby the cost efficiency is no more than 20% of that incurred by optimal MEC resource allocation. In addition, our impact factor analysis indicates that MEC applications with extreme low latency requirements (e.g., 10ms) are more in favour of small size servers for cost efficiency purposes.

Chapter 5

Fault-Tolerant End-to-End Resource Management

Having studied the minimization problem of deployment cost and operational cost for low latency networks, in this chapter, we focus on enhancing the fault tolerance of cost-efficient low latency networks. The objective is to further guarantee both cost efficiency and low latencies in face of network failures. In particular, different end-to-end resources such as active and back-up computing/bandwidth resources from on-path VNFs (i.e., components of service function chains) and network links will be jointly optimized. As such, we complement the studies carried out in Chapter 4, whereby only routing resources and stateless computing resources at end services are jointly optimized.

5.1 Introduction

Cloud service provides exploit different network functions (NFs), such as network address translation (NAT), firewall and deep packet inspection (DPI), to improve network performance and security. These NFs are embedded into dedicated hardware that are costly and difficult to reconfigure. The advent of NFV provides a more flexible and inexpensive support of NFs compared to conventional hardware-based approaches [8]. Specifically, NFV decouples NFs from physical devices by implementing NFs as software running in VMs in the form of VNFs. As such, VNFs can be instantiated on any DC with available computing resources. This flexibility in VNF instantiation further enables advanced VNF placement schemes [140], through which the cost and performance of NFs can be largely improved [141].

Despite the achieved flexibility, moving NFs from hardware to software poses grand concerns especially in terms of reliability. For instance, VNFs are software running in DCs, which are vulnerable to various problems such as misconfiguration, faulty VMs and software malfunctions [142, 143]. In order to enhance VNF fault tolerance, backup VNFs that run in stand-by instances are required [144]. In case of failures, requests to stateless VNFs can be immediately redirected to one of their stand-by instances. In contrast, stateful VNFs generate states during traffic processing [145] that need to be transferred to stand-by instances in order to guarantee seamless request redirection. For instance, a stateful NAT VNF needs to maintain existing user connections to support its correct operation. If a NAT fails, the transient states created by the traffic itself have to be transferred to the backup NAT to avoid NAT disconnection. Given that such state transfers need to be continuously performed while active instances are in operation [146, 90], it could consume considerable network bandwidth resources, and lead to significant network link overheads. Furthermore, if the network path used for state transfers overlaps with the VNF request routing path, the active VNF instance's request admissions may fail due to delay violations caused by link congestion. As such, decisions regarding 1) the placement of active instances, 2) the placement of stand-by instances, 3) request routings, 4) the state transfer paths need to be jointly considered so that the number of admitted user requests can be maximized. In this paper, we study the *fault-tolerant stateful VNF placement problem*, whereby the aforementioned four decisions are jointly determined under DC computing and bandwidth resource constraints.

Providing efficient solutions to the fault-tolerant VNF placement problems poses several challenges. On one hand, as stated earlier, a naive solution that separately determines the instance locations and routings may result in network congestion and admission failures. It may also lead to significant network communication costs if the active/stand-by instances are placed with long network distance to the source and destination nodes of requests. On the other hand, the number and placements of stand-by instances directly influence the state update cost for VNFs. At the same time, the number of stand-by instances affects the robustness of the networks. Clearly, a higher number of stand-by instances indicates a higher degree of fault tolerance.

Previous studies on the fault-tolerant VNF placement problem have either focused on backup instances or stateless VNFs [93, 144, 147, 148, 149]. For example, [93] investigated the planning-stage VNF backup instances (i.e., do not consider active instances) deployment problem while taking into account the failure probabilities of network nodes. [148] studied the placement problem of redundant stateless VNFs in LTE networks with a focus on deriving the optimal number of VNFs to guarantee reliability. [144] investigated the joint active and backup stateless VNF placement problem, but did not consider request routing and VNF state transfers. To the best of our knowledge, this work is the first study that jointly considers stateful active/stand-by VNF placement, request routing and state transfers.

In the remainder of this paper, we first introduce the considered scenario and the related definitions in Section 5.2. Then, we propose an efficient heuristic based on the joint availability of DC computing resources and the accumulative bandwidth resources of DC's inbound links in Section 5.3. The proposed heuristic jointly computes the placement of both active and stand-by stateful VNF instances. For a special case of our problem without bandwidth constraint, we propose a $(2, 4+\varepsilon)$ bicriteria approximation algorithm with proved approximation ratios on the achieved cost and maximum DC utilization in Sections 5.4. The proposed algorithm exploits an approach based on auxiliary graph that allows active/stand-by instances, request routings and state update paths to be jointly considered. The evaluation results presented in Section 5.5 suggest that the proposed algorithms significantly improve the request admission rate while reducing DCs' cost. At the same time, they outperform existing solutions that separately consider placements, routings and update paths. Concluding remarks are finally presented in Section 5.6.

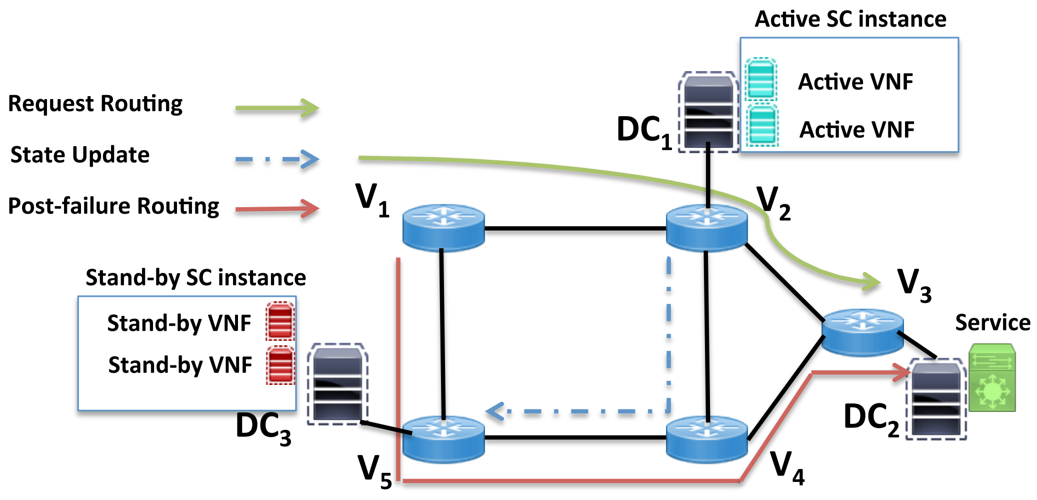


Figure 5.1: An example of fault-tolerant placement problem in G with a set $\mathscr{DC} = \{DC_1, DC_2, DC_3\}$ connected by a set $V = \{v_2, v_3, v_5\}$ of switches.

5.2 Fault-Tolerant Virtual Network Function Placement Problem

5.2.1 System Model

We model the network $G = (V \cup \mathscr{DC}, E)$ operated by a cloud service provider with a set V of switches, a set $\mathscr{DC}$ of DCs attached to V, and a set E of network links (see Fig 5.1). We follow the convention to assume that the number of DCs is far less than the number of switches. Each $DC_i \in \mathscr{DC}$ has computing resources $C(DC_i)$ that can be utilized to instantiate VNFs instances. A sequence of VNFs forms a *service chain*, denoted as SC, and *an instance of a service chain* is defined as an implementation of its specified VNFs in a VM. Given the computing capacity $C(DC_i)$ of DC_i , a limited number of instances of different service chains can be supported in each DC. Similarly, each link $e \in E$ has a capacity $B(e)$ of bandwidth resources that can be allocated to user requests. Without loss of generality, we assume that each DC and the switch node attached to it is connected by a high-speed optical cable with abundant network bandwidth (see Fig 5.1) so that the delay and communication cost incurred at these links can be considered as negligible. Furthermore, the transmission delay on each link $e \in E$ is denoted as d_e.

5.2.2 Requests for Service Chains

We denote as $r_j = (s_j, t_j; SC_j, \rho_j, D_j)$ user request j. Each user request requires to be routed from a source node s_j to a destination node t_j at a given packet rate ρ_j within D_j time, such that its traffic passes through one instance of its required service chain SC_j.

Different user requests have different demands for SC, with each type of service chains having a different sequence of VNFs. Without loss of generality, we assume that the computing resource requested by an instance of service chain SC_j for processing the traffic of r_j is proportional to its packet rate, i.e., $\rho_j \cdot c_b$, where c_b is a given constant representing the amount of computing resources that is needed to process each unit packet rate. The total amount of computing resource allocated to all instances of service chains in a data center DC_i must not exceed its computing capacity $C(DC_i)$.

The *end-to-end delay requirement* D_j of each request r_j specifies the maximum tolerable delay experienced by its traffic from its source node, s_j, to its destination node, t_j. It consists of the processing delay of service chain SC_j at a DC and the transfer delay on each link. Specifically, assuming an instance of SC_j at DC_i is assigned to process the traffic of r_j, then its experienced delay consists of the transfer delay $d(s_j, DC_i)$ from s_j to DC_i, the processing delay $d(SC_j, DC_i)$ by an instance of SC_j at DC_i, and the transfer delay $d(DC_i, t_j)$ from DC_i to t_j. The end-to-end delay requirement of r_j is:

$$d(s_j, DC_i) + d(SC_j, DC_i) + d(DC_i, t_j) \le D_j. \tag{5.1}$$

5.2.3 Stateful Active and Stand-by Instances

Faults can occur anywhere and at anytime in a network due for example to natural disasters in the locations of DCs, software malfunctions in VNFs, and hardware failures in DCs. To avoid service interruption due to such failures, we assume that an *active instance* of service chain of each request is placed into one DC, and a few *stand-by instances* of the service chain are placed into other DCs. For simplicity, the instances are considered at the service chain level which consists of various VNFs. Once the active instance fails (e.g., one of the composite VNFs within a SC fails), its traffic can be seamlessly redirected to one of the stand-by instances for processing. In this work, we consider *stateful* VNFs (i.e., stateful SCs), whereby the states from the active instance need to be constantly transferred to stand-by instances while the active instance is still in operation. Such state transfer plays a vital role in enabling the seamless and correct request redirection from an active stance to a stand-by instance.

We denote as DC_j^a the DC where the active instance of service chain SC_j of user request r_j is placed and denote as $\mathscr{DC}_j^s$ the set of DCs where stand-by instances of SC_j are placed. We assume that the state update rate of each request from its active instance to stand-by instances is proportional to its packet rate, i.e., $\beta \cdot \rho_j$, where β (> 0) is a given constant. We further assume that the computing resource demand of stand-by instances will be allocated only when they are activated (i.e., stand-by instances do not consume computing resources in the placement problem). Since the focus of our work is on the pre-failure placement of active/stand-by VNFs, we consider the resource provisioning of back-up instances (after VNFs fail) out of the scope of this paper.

5.2.4 Cost Model

Minimizing the implementation cost for user requests is usually considered as an effective objective to reduce the operational cost of network service providers. Here, *the implementation cost* of a request $r_j = (s_j, t_j, SC_j, \rho_j, D_j)$ consists of (i) the operational cost of computing resource to process requests, i.e., the use of an active instance of service chain SC_j in DC_j^a, (ii) the communication cost of transferring its traffic from s_j to DC_j^a for processing, (iii) the communication cost of transferring the processed data from DC_j^a to its destination t_j, and (iv) the communication cost of updating status from DC_j^a to DCs in $\mathscr{DC}_j^s$. Let $c(SC_j, DC_i)$ be the cost of implementing an instance of service chain SC_j of r_j in DC_i, and $c(e)$ be the cost of transferring a unit packet rate for request r_j through link $e \in E$. Without loss of generality, we assume that the edge cost $c(e)$ is within the range of $(0, 1]$. Then, the implementation cost $c(r_j)$ of r_j in active DC_j^a and a set $\mathscr{DC}_j^s$ of stand-by DCs of the network is:

$$\begin{aligned} c(r_j) = \rho_j \Bigg(c(SC_j, DC_j^a) + \sum_{e \in p_{(s_j, DC_j^a)}} c(e) + \sum_{e \in p_{(DC_j^a, t_j)}} c(e) \\ + \sum_{DC_i \in \mathscr{DC}_j^s} \sum_{e \in p_{(DC_j^a, DC_i)}} c(e) \Bigg), \end{aligned} \tag{5.2}$$

where $p_{(y,z)}$ is the shortest path in G from node y to node z.

5.2.5 Problem Definition

Different cloud service providers may have different network performance indicators to optimize the service delivery process of their networks. To cater for the different optimization objectives of different network service providers, we study two different fault-tolerant VNF placement problems that correspond to different operators' needs as follows.

1) Considering that start-up service providers have limited computing and bandwidth resources, their main interest is to admit as many requests as possible, so that their limited resources are perfectly utilized while achieving the least operational cost. Thus, we consider the optimization objective as the maximization of the admitted number of requests. Specifically, the goal of *the fault-tolerant VNF placement problem* is for all user requests r_j in $\mathscr{R}$ to place an active instance of service chain SC_j to a DC_j^a, to place a number of stand-by instances to a set of DC_j^s, to find the routing path for requests from s_j to t_j via DC_j^a and to find the state update path from DC_j^a to DC_j^s, so that as many requests as possible are admitted while the total cost of implementing these admitted requests is minimized, subject to the computing resource capacity constraints $C(DC_i)$, the network bandwidth capacity $B(e)$ for $e \in E$, and the end-to-end delay constraints.

2) Service providers that provide computing-intensive workload processing in distributed DCs may want their DCs to be balanced (e.g., geographical load balancing), such that users in different locations have maximum resource availabilities with guaranteed user experiences. Assuming that links in G have abundant

resources to implement all requests in $\mathscr{R}$, let $\mathscr{R}_i$ be the set of instances of service chains that are admitted by DC_i. The goal of *the fault-tolerant VNF placement problem without bandwidth capacity constraint* is the same as the fault-tolerant VNF placement problem except that the objective is to minimize the maximum DC utilization for all DCs, i.e.,

$$\min \max_{DC_i \in \mathscr{D}\mathscr{C}} \sum_{r_j \in \mathscr{R}_i} \frac{\rho_j \cdot c_b}{C(DC_i)}, \tag{5.3}$$

while the cost of implementing all requests is minimized, i.e.,

$$\min \sum_{DC_i \in \mathscr{D}\mathscr{C}} \sum_{r_j \in \mathscr{R}_j} c(r_j), \tag{5.4}$$

subject to the computing resource capacity constraints of DCs in $\mathscr{D}\mathscr{C}$ and the end-to-end delay constraints of requests.

Both problems are clearly NP-hard given that a special version of these problems without considering fault-tolerant requirements and (or) bandwidth resource constraints is NP-hard by simple reduction from another NP-hard problem, the unsplittable single-source flow problem [150].

5.3 Fast Heuristic Solution for Fault-Tolerant Placement Problem

Due to the NP-hardness of the problem, in the following, we propose an efficient heuristic to solve it.

5.3.1 Algorithm

To avoid poor performance in terms of request admission rate and cost, the placement of active/stand-by instances, request routings and update paths need to be jointly computed. Conventional approaches such as naive *greedy algorithm* select DCs for the active and stand-by instances separately. It first finds the DC with the largest amount of available computing resources to host the active instance for r_j, and then selects a random number of DCs with lowest transfer costs to host stand-by SC instances for r_j. As a result, the separate placement and routing decision may result in situations where no update paths are available from the active instance to one of its stand-by instances due to link congestions.

In contrast, our heuristic jointly selects a DC for the active instance and a number of DCs for its stand-by instances. Specifically, the heuristic first sorts all requests in $\mathscr{R}$ in increasing order of their rates, and then sequentially considers the requests in the sorted list. Next, for the jth request r_j in the sorted list, the algorithm ranks DCs based on the increasing order of the product of the available computing resources and the accumulative available network bandwidth resources of DC's inbound links. Let $NR(DC_i, j)$ be the ranking of DC_i after considering the $(j-1)$th request in the sorted list. Let also denote $A(DC_i, j)$ and $A(e, j)$ as the available computing and bandwidth resources of DC_i and link e after considering the $(j-1)$th request. Then,

$$NR(DC_i, j) = A(DC_i, j) \cdot \sum_{e \in E^i_{adj}} A(e, j), \tag{5.5}$$

where E^i_{adj} is the set of inbound links of DC_i. The idea of such ranking is to find a set of DCs that not only have enough computing but also network bandwidth resources for both active and stand-by instances.

Based on the obtained ranking, the algorithm selects the DC with the highest rank, denoted DC_{hr}. Then, the algorithm checks if (1) DC_{hr} has enough computing resources to host an active instance of SC_j for r_j; and (2) if the shortest path from s_j to t_j via DC^a_j has enough bandwidth resources to transfer r_j at rate ρ_j. The algorithm also checks whether (3) DC_{hr} conforms to r_j's delay requirement. If the above three requirements are all satisfied, DC_{hr} is selected as DC^a_j. The algorithm then searches stand-by instances for r_j. To this end, the rest of DCs other than DC_{hr} are sorted in the increasing order of state update costs to DC_{hr}. Each DC in the sorted DC list is further added to $\mathscr{DC}^s_j$ until there is a DC that cannot meet the bandwidth resource requirement for updating states from DC_{hr}. To avoid all the other data centers to be selected to host stand-by instances, we set a threshold K ($1 \leq K \leq |\mathscr{DC}|$) for the number of DCs that can be used for stand-by instances. This prevents a large number of DCs to be selected to place stand-by instances and as such avoids the creation of unnecessary burden for state updates. If no stand-by DC exists after considering the rest of the DCs, request r_j is rejected.

In case constraints (1), (2) and (3) cannot be satisfied, DC_{hr} is added to $\mathscr{DC}^s_j$ as the accumulative bandwidth resources to nearby DCs might make DC_{hr} a promising candidate for stand-by instances. DCs other than DC_{hr} are sorted in a list L_{hr} based on the increasing accumulative communication cost to DC_{hr}. The algorithms then iterates through DCs in L_{hr} until a DC_i that can serve the active service chain instance is found, i.e., a DC that meets constraints (1), (2) and (3). Once such a DC_i is found, it is selected to be the DC that hosts the active instance of r_j. Among the rest DCs in L_{hr}, only the ones that have enough bandwidth resources for state updates rate $\beta \cdot \rho_j$ from DC^a_j are added to $\mathscr{DC}^s_j$ (with $|\mathscr{DC}^s_j| \leq K$). If neither such DC can be found for its active instance nor a set of DCs can be determined for its stand-by instances, r_j is rejected.

The above procedure continues until all requests in $\mathscr{R}$ are considered. The details of the proposed heuristic are shown in Algorithm 7.

5.3.2 Algorithm Complexity

The performance of the proposed heuristic is given by the following theorem.

Theorem 4. *Given a network $G = (V \cup \mathscr{DC}, E)$, let $\mathscr{R}$ be a set of requests with each represented by $r_j = (s_j, t_j, SC^k, \rho_j, D_j)$. Algorithm 7 delivers a feasible solution to the fault-tolerant VNF placement problem in $O(|\mathscr{R}|(|\mathscr{DC}|\log|\mathscr{DC}|) + (|V| + |\mathscr{DC}|)^3)$ time.*

Proof. To show the feasibility of the algorithm, we need to show that the resource demands of each admitted request and its end-to-end delay requirement are met. Clearly, this is true due to steps 8 and 24.

For the running time of the proposed heuristic, we can see that the most time-consuming phases of Algorithm 7 are (1) finding all pair shortest paths in G, (2)

Algorithm 7 `Heuristic`

Input: Network $G(V \cup \mathscr{DC}, E)$; Set of requests $r_j \in R$ where $r_j = (s_j, t_j, SC_j, \rho_j, D_j)$, K.

Output: Assignments of each request in $r_j \in \mathscr{R}$ to DC_j^a for active SC instances and to $\mathscr{DC}_j^s$ for stand-by SC instances.

```
1: for $r_j \in \mathscr{R}_j$ do
2:     $Sorted_{list} \leftarrow$ SortIncreaseOrder($\mathscr{DC}$) based on Eq. (5.5)
3:
4:     $DC_{hr} \leftarrow Sorted_{list}.getFirst()$;
5:     $\mathscr{DC}_j^s \leftarrow \emptyset$ and $DC_j^a \leftarrow NIL$;
6:     $A(p_{(s_j, DC_{hr})}) \leftarrow G.shortestPathAvailBandwidth(s_j, DC_{hr})$;
7:     $A(p_{(DC_{hr}, t_j)}) \leftarrow G.shortestPathAvailBandwidth(DC_{hr}, t_j)$;
8:     if $\rho_j \le A(p_{(s_j, DC_{hr})})$ && $\rho_j \le A(p_{(DC_{hr}, t_j)})$ && $D_{hr} \le D_j$ then
9:         $DC_j^a \leftarrow DC_{hr}$;
10:        $Update_{list} \leftarrow$ SortIncreaseOrder($\mathscr{DC} \setminus DC_{hr}$) based on state update costs to $DC_{hr}$;
11:        for each $DC_i \in L_{hr}$ do
12:            $\mathscr{DC}_j^s \leftarrow \mathscr{DC}_j^s \cup \{DC_i\}$
13:            if $K = |\mathscr{DC}_j^s|$ or $A(p_{(DC_i, DC_{hr})}) \le \beta \cdot \rho_j$ then
14:                Break;
15:            end if
16:        end for
17:    else
18:        $\mathscr{DC}_j^s \leftarrow \mathscr{DC}_j^s \cup \{DC_{hr}\}$
19:        $L_{hr} \leftarrow$ SortIncreaseOrder($\mathscr{DC} \setminus DC_{hr}$) following state update costs to $DC_{hr}$;
20:        for each $DC_i \in L_{hr}$ do
21:            if $DC_j^a \ne NIL$ && $A(p_{(DC_j^a, DC_i)}) \ge \beta \cdot \rho_j$ && $|\mathscr{DC}_j^s| \le K$ then
22:                $\mathscr{DC}_j^s \leftarrow \mathscr{DC}_j^s \cup \{DC_i\}$;
23:            else
24:                if $\rho_j \le A(p_{(s_j, DC_i)})$ && $\rho_j \le A(p_{(DC_i, t_j)})$ && $D_i \le D_j$ then
25:                    $DC_j^a \leftarrow DC_i$;
26:                else
27:                    if $|\mathscr{DC}_j^s| \le K$ then
28:                        $\mathscr{DC}_j^s \leftarrow \mathscr{DC}_j^s \cup \{DC_i\}$;
29:                    end if
30:                end if
31:            end if
32:        end for
33:    end if
34:    Update DCs' available resources and network link resources
35: end for
    return The assigned DC to place the service chain of each request for the processing of its traffic, and a set of DCs to replicate its service chain.
```

ranking all DCs, and (2) iteratively selecting a number of DCs for each request. Clearly, phase (1) takes $O((|V| + |\mathscr{DC}|)^3)$ time, phase (2) takes $O(|\mathscr{DC}| \log |\mathscr{DC}|)$ time, and phase (3) takes $O(|\mathscr{DC}|)$ time. Since the ranking of DCs is performed every time when a request is admitted, the overall running time of algorithm 7 is $O(|\mathscr{R}|(|\mathscr{DC}| \log |\mathscr{DC}|) + (|V| + |\mathscr{DC}|)^3)$. □

5.4 Approximate Solution for a Special Problem Instance

In this section, we consider the fault-tolerant VNF placement problem without bandwidth capacity constraint. We assume that the network G has enough bandwidth resources on its links, and all requests in $\mathscr{R}$ can be admitted. For this problem, we propose a bicriteria approximation algorithm with an approximation ratio of $(2, 4+\varepsilon)$. Such a ratio indicates that (1) the implementation cost of all requests is twice the optimal cost, and (2) the minimum maximum utilization of computing resources in a DC is $(4+\varepsilon)$ times the optimal one, where ε is a constant with $\varepsilon > 0$.

5.4.1 Overview

Given network G and a set $\mathscr{R}$ of requests, the fault-tolerant VNF placement problem without bandwidth capacity constraint is to balance the workloads among DCs by not only minimizing the maximum resource utilization of DCs but also minimizing the total requests' implementation costs. One challenge is with respect to the tradeoff between the balance of DC utilizations and the implementation costs of requests. For instance, the active instance of some requests may have to be placed into DCs with high communication costs in order to achieve a balanced workload among DCs. In order to achieve a near optimal solution, we jointly consider the active/stand-by instance placements, request routings and state update paths.

The idea behind the proposed approach is to reduce the fault-tolerant NFV placement problem without the bandwidth capacity constraint in G into a single-source unsplittable flow problem [150] in an auxiliary graph $G' = (V', E')$. Then, a feasible unsplittable flow in G' that minimizes both the implementation cost of requests and the maximum congestion of links in G' is a feasible solution to the original problem in G. Note that the aim of the single-source unsplittable flow problem is, given a network $G = (V, E, u)$, a source vertex s, and a set of M commodities with sinks t_1, ..., t_M and associated real-valued demands σ_1, ..., σ_M, to route the demand σ_m of each commodity m along a single $s - t_m$ flow path so that the congestion, i.e., $max_{e \in E}\{\frac{f_e}{u_e}, 1\}$, and the cost of flow f are minimized, while the edge capacities constraints of G are met. To solve the unsplitable flow problem, Kolliopoulos and Stein [150] gave $(2,4+\varepsilon)$ approximation algorithm for flow cost and link congestion.

5.4.2 A $(2,4+\varepsilon)$ Bicriteria Approximation Algorithm

The approximation algorithm first constructs the auxiliary graph $G' = (V', E')$. Recall that the traffic of each request r_j is processed by an active instance of its SC_j in a DC, and by one of its stand-by instances in other DCs if the active instance fails. Thus, each DC_i corresponds to a *DC node* (see Fig. 5.2), and is added into the auxiliary graph G', i.e., $V' \leftarrow \{DC_i \mid 1 \leq i \leq |\mathscr{DC}|\}$. For each DC node DC_i, we further add a *virtual DC node* DC_i' (see Fig. 5.2) into V', i.e., $V' \leftarrow V' \cup \{DC_i'\}$ so that the DC capacity constraint is converted into a link constraint. Next, for each DC node, we add a few *stand-by set nodes* to G', whereby each stand-by set node represents a candidate set of DCs for stand-by instances (see Fig. 5.2). Specifically, the stand-by set nodes of DC_i are different combinations of DCs from $\mathscr{DC} \setminus \{DC_i\}$ whereby each stand-by set node has no more than K DCs. For example, DC_1 from Fig. 5.2 has a 3 stand-by set nodes DC_2, DC_3 and node DC_2,DC_3 where $k = 2$. Note that a stand-by set node will not be added twice (e.g., there is only one DC_1 ins stand-by set nodes). Last, we add a *request node* into V' for each request r_j, and add a common source s_0 for all requests into V'.

An edge from the common source s_0 to each of stand-by set node is added into E'. Its capacity and cost are set to infinity and zero, respectively (i.e., no bandwidth constraint). Also, there is an edge from each stand-by set node to a DC node DC_i if DC_i is not in the set of DCs represented by the stand-by set node (e.g., DC_1 has edges to DC_2, DC_3 and DC_2 & DC_3 in Fig. 5.2). The capacity of the edge is set to infinity, and its cost is the accumulative cost of state updates from DC_i to the DCs within the set of DCs represented by the stand-by set node. Further, an edge from DC_i to DC_i' is added. Its capacity is the processing capacity of DC_i, and its cost is set to 0. We add an edge from each DC_i' to a request r_j if DC_i provides a total delay (e.g., sum of processing and communication delay) for request r_j smaller than the request delay requirement. The capacity of this edge is set to infinity. Its cost is

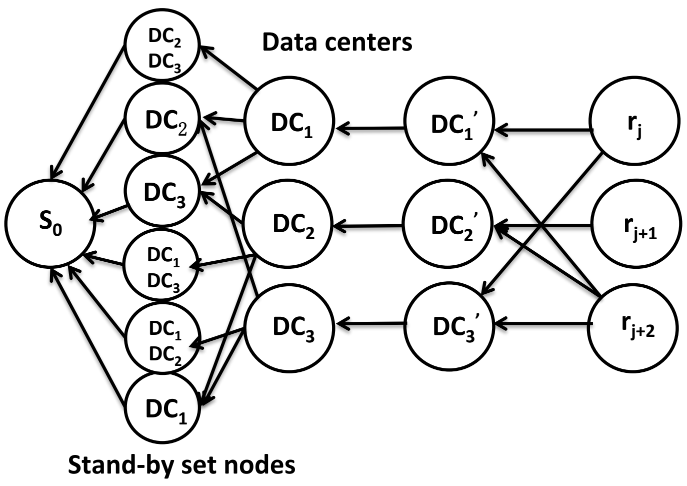


Figure 5.2: An example of the auxiliary graph $G' = (V', E')$ constructed from network G with a set $\mathscr{DC} = \{DC_1, DC_2, DC_3\}$ of DCs that are connected by a set $V = \{v_2, v_3, v_5\}$ of switches. $\mathscr{R} = \{r_j, r_{j+1}, r_{j+2}\}$.

Algorithm 8 An $(2, 4+\varepsilon)$ Bicriteria Approximation Algorithm for the fault-tolerant VNF Placement Problem without Network Bandwidth Constraint

```
Input: A network $G(V \cup \mathscr{DC}, E)$, a set requests $r_j \in R$ where $r_j = (s_j, t_j; SC_j, \rho_j, D_j)$.
Output: Assignments of each requests in $r_j \in \mathscr{R}$ to $DC_j^a$ for active SC instances and to $\mathscr{DC}_j^s$ for
    stand-by SC instances.
1: Construct an auxiliary graph $G' = (V', E')$ from network $G(V \cup \mathscr{DC}, E)$ as exemplified by Fig.
   5.2;
2: Find a single-source unsplittable flow$f$ in the auxiliary graph $G'$ by applying the algorithm
   presented in [150];
3: The requests that are assigned into $DC_i$ in the flow $f$ will be processed by an instance of a service
   chain in $DC_i$, and request will be assigned a set of DCs that are represented by the stand-by set
   node in $f$.
       return The assigned DC to place the service chain of each request for the processing of its
   traffic, a set of DCs to replicate its service chain, the request routings and update paths.
```

the total cost of processing costs of DC_i for request r_j (e.g., $\rho_j c(SC_j, DC_j^a)$) plus the communication costs from s_j to DC_i and from DC_i to t_j at packet rate ρ_j. Fig. 5.2 shows an example of the constructed auxiliary graph G'.

Given the constructed auxiliary graph $G'(V', E')$, the original problem is transferred to the problem of single source unsplittable flow problem in G'. To find a feasible flow f in G', the algorithm presented in [150] is invoked. The main steps of the approximation algorithm are shown in Algorithm 8.

5.4.3 Algorithm Analysis

We now analyze the correctness and performance of the proposed algorithm.

Theorem 5. *Given a network $G = (V \cup \mathscr{DC}, E)$, let $\mathscr{R}$ be a set of requests with each represented by $r_j = (s_j, t_j, SC^k, \rho_j, D_j)$). Algorithm 8 delivers a bicreteria approximate solution with an approximation ratio of $(2, 4+\varepsilon)$ with (1) the implementation cost of all requests twice the optimal cost, and (2) the minimum maximum utilization of computing resource in a DC $(4+\varepsilon)$ times the optimal one, for the fault-tolerant VNF placement problem without bandwidth capacity constraint, in $O(T_2(|\mathscr{R}|+|V|+|\mathscr{DC}|\binom{|\mathscr{DC}|-1}{k}, |\mathscr{R}|\cdot|\mathscr{DC}|+|\mathscr{DC}|\binom{|\mathscr{DC}|-1}{k})))$ time, where $T_2(m,n)$ is the time to solve a fractional minimum-cost flow problem with m edges and n nodes in the flow graph, and ε is a constant with $\varepsilon > 0$.*

Proof. We first show the feasibility of the proposed algorithm. Given an unsplittable flow f, it starts at a request node r_j and ends at the common source s_0 in G' according to the construction of auxiliary graph G'. Clearly, a DC node DC_i for active instance and a stand-by set node exists in the route. The traffic of request r_j is processed by the placed active instance in DC_i, and it is routed on the paths from r_j's source s_j to DC_i and from DC_i to destination t_j (e.g., represented by edge $\langle DC_i', r_j\rangle$ in auxiliary graph). Also, since an auxiliary edge between a stand-by node to DC_i denotes the state update path from DC_i to one of the stand-by set nodes, the processing states are then updated to one of the stand-by set nodes following the traversed edge by f. In addition, the delay requirement of r_j is met, as f only exists when there is an edge between r_j and DC_i' (i.e., delay is met).

We then show the approximation ratio of the devised approximation algorithm. It is clear that the solution to the single-source unsplittable flow problem in auxiliary graph G' corresponds to the solution to the VNF placement problem without bandwidth constraint in network G. The approximation ratio obtained for the former problem thus is the approximation ratio for the latter, i.e., $(2, 4+\varepsilon)$.

We then show the time complexity of the approximation algorithm, which can be divided into two stages: (1) the construction of the auxiliary graph $G'(V', E')$; and (2) finding an unsplittable flow in the constructed auxiliary graph using the algorithm proposed by Kolliopoulos and Stein [150]. Clearly, the construction of G' takes $(|V'|+|E'|)$ time, where $|V'| = O(|\mathscr{R}|+|V|+|\mathscr{DC}|+\sum_{k=1}^{|\mathscr{DC}|-1}\binom{|\mathscr{DC}|-1}{k}) =$

$O(|\mathscr{R}| + |V| + |\mathscr{DC}|\binom{|\mathscr{DC}|-1}{k})$, and $E' = O(|\mathscr{R}| \cdot |\mathscr{DC}| + |\mathscr{DC}|\binom{|\mathscr{DC}|-1}{k})$, where $\sum_{k=1}^{|\mathscr{DC}|-1}\binom{|\mathscr{DC}|-1}{k}$ is the maximum number of stand-by set nodes for all DCs. According to [150], finding a unsplittable flow in G' takes $O(T_2(|V'|,|E'|) + |E'|\log(|V'|/\varepsilon)) = O(T_2(|\mathscr{R}| + |V| + |\mathscr{DC}|\binom{|\mathscr{DC}|-1}{k}, |\mathscr{R}| \cdot |\mathscr{DC}| + |\mathscr{DC}|\binom{|\mathscr{DC}|-1}{k}))$ time. □

5.5 Evaluations

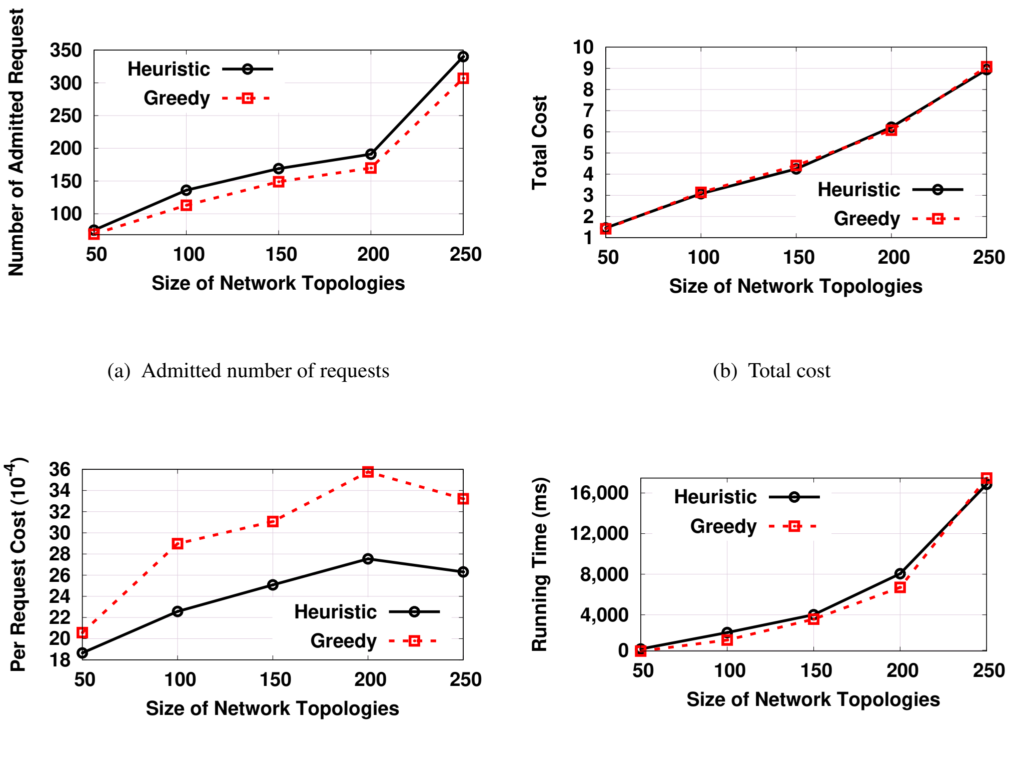
5.5.1 Experiment Settings

We consider synthetic networks generated by GT-ITM [151]. The network size ranges from 50 to 250 nodes with a node connectivity of 0.2 (i.e., the probability of having an edge between two nodes is 0.2) [24]. In these networks, the server to DC ratio is set to 0.1, and each DC has a CPU capacity in the range 4,000 to 8,000 Mhz. The transmission delay of a network link varies between 2 milliseconds (*ms*) and 5 *ms* [152]. The costs of transmitting and processing 1 GB (approximately 16,384 packets with each having size of 64 *KB*) of data are set within [\$0.05, \$0.12] and [\$0.15, \$0.22], respectively, following typical charges in Amazon EC2 with small variations [153]. We consider five categories of NFs: Firewall, Proxy, NAT, DPI, and Load Balancer, their computing demands (e.g., CPU) are adopted from []. Further, the consumed computing resources of a service chain is the sum of the computing demands of its contained NFs (the number of contained NFs is randomly selected between 1 and 50). The processing delay of a packet for each NF is randomly drawn from 0.045 *ms* to 0.3 *ms* [127], and the processing delay of a service chain is the total processing delay of its NFs. Each request r_j is generated by randomly selecting its source s_j and destination t_j from G with packet rate ρ_j randomly selected between 400 and $4,000$ packets/second [154]. Each request has a delay requirement ranging from 10 *ms* to 100 *ms* [155, 23]. The running time is obtained based on a machine with a 3.40GHz Intel i7 Quad-core CPU and 16 GB RAM.

There has not been any existing work considering the fault-tolerant stateful VNF placement. One possible solution is to derive each decision variable in a separate step (similar to an existing approach for stateless VNF placement problem [144]). In this sense, we compare our algorithms against a greedy algorithm (described in 5.3.1) that separately selects the placement of active/stand-by SC instance, request routings and state transfer paths. The greedy aims to maximize the throughput by admitting requests with small packet rates first. For simplicity, we refer to this greedy algorithm as algorithm `Greedy`, and the greedy without bandwidth constraint as $Greedy_{noBW}$. The proposed heuristic and approximation algorithms (Algorithms 7 and 8) are referred to as `Heuristic` and `Appro`, respectively.

5.5.2 Performance Evaluation

We first compare the performance of algorithm `Heuristic` against that of algorithm `Greedy` for networks with various sizes. Fig. 5.3 shows the result in terms of the number of admitted requests, the average cost of admitting a request, and the running time. We see from Fig. 5.3 (a) that the proposed algorithm `Heuristic` consistently achieves a number of admitted requests higher than `Greedy` by 10%. This is due to the fact that algorithm `Heuristic` jointly selects the active and stand-by instances. As such, both network resources and DCs' computing resources are efficiently utilized, which avoids the request rejections that happened with `Greedy` due to separate selection process. The surge of admitted requests observed with both algorithms for networks with size equal to 250 can be explained by the fact that in this case, the bandwidth resources between any two network nodes are on average increased (i.e., more network links exist between two nodes when network size becomes larger), which results in relaxed constraints in terms of bandwidth. From Fig. 5.3 (b), we observe that the two algorithms achieve almost the same total cost. However, since the overall admitted request number obtained with `Heuristic` is higher than that of `Greedy`, we see from Fig. 5.3 (c) that the `Heuristic` achieves a lower per request cost than `Greedy`. Furthermore, `Heuristic` achieves a lower cost in terms of the average cost per admitted request than that of `Greedy`. Meanwhile, we see from Fig. 5.3 (d) that `Heuristic` slightly results in a longer running time than that of algorithm `Greedy`.



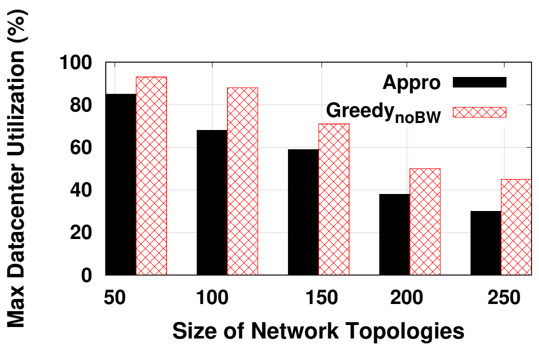
(a) Admitted number of requests (b) Total cost (c) Average cost per request (d) Running time

Figure 5.3: Performance of algorithms `Heuristic` and `Greedy`.

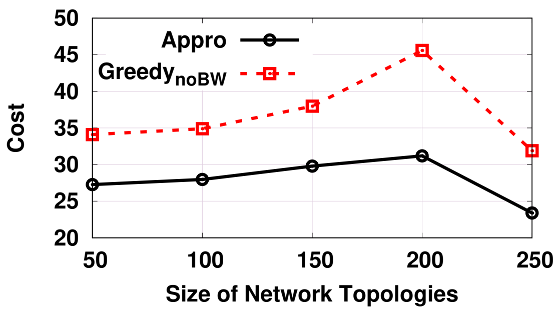
We then compare the performance of algorithm `Appro` with that of algorithm $Greedy_{noBW}$ in terms of the maximum resource utilization of DCs, the average cost of implementing a request, and the running time under the same network settings. It can be seen from Fig. 5.4 (a) that the proposed algorithm `Appro` consistently delivers solutions with lower maximum DC resource utilization than that obtained with algorithm $Greedy_{noBW}$. For example, when the network size is 100, the minimum maximum resource utilization of DCs of `Appro` is 10% lower than that of algorithm $Greedy_{noBW}$. The rationale behind is that algorithm `Appro` explores a fine-grained trade-off between the resource utilizations and the cost of implementing requests. Fig. 5.4 (a) also shows that the maximum resource utilization of DCs is decreasing with the network size. This is because larger networks mean on average more computing resources in DCs, which incurs lower resource utilization. In addition, as shown in Fig. 5.4 (b) and (c), algorithm `Appro` also delivers a lower implementation cost. Regarding the running time, it should be noted that our algorithms are intended to be executed offline and to compute solutions that will be implemented in DC networks at the network configuration stage. The running time of `Appro` is therefore considered as tolerable. Finally, we observe that both the maximum DC utilization in Fig. 5.4 (a) and the total cost in Fig. 5.4 (b) are not increasing as the network size grows, which further justifies the performance guarantee of the proposed `Appro` algorithm.

5.6 Conclusion

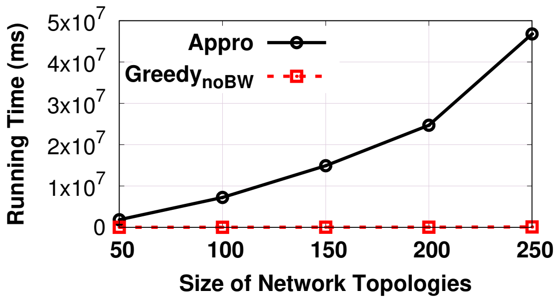
In this chapter, we proposed a novel efficient heuristic approach to enhance the fault tolerance of cost-efficient low latency communications, which jointly computes the placement of active and stand-by instances of stateful VNFs, the routing paths and update paths of user requests. For a special case of the problem without network bandwidth constraint, we proposed a bicriteria approximation algorithm with performance guarantees. We evaluated the performance of the proposed algorithms based on simulations under realistic settings. Our evaluation results show that the performance obtained with each algorithm outperforms existing solutions that separately determine placements, routings and state update paths.



(a) Maximum DC utilization



(b) Total cost



(c) Running time

Figure 5.4: Performance of algorithms `Approximation` and *Greedy*$_{noBW}$.

Chapter 6

General Conclusions

6.1 Summary

In the upcoming 5G era, low latency has been defined as the essential feature to realize the emerging innovative applications such as autonomous car control, AR and remote surgery. The current practice to achieve low latency is to overprovision bandwidth and computing resources. However, this approach results in resource wastage, and is not applicable to large-scale networks due to cost issues. To make low latency applications accessible to the general public, ISPs will need to design their low latency networks in a cost-efficient way.

In this thesis, the problem of designing cost-efficient low latency networks has been addressed. In particular, we proposed a cost-efficient resource management framework that solved 1) the cost-efficient design of low latency communication infrastructures; 2) the cost-efficient design of dynamic resource management for low latency applications; and 3) the cost-efficient design of fault-tolerant resource management.

In Chapter 3, three network planning algorithms for cost-efficient low latency networks have been presented, whereby the resulting communication infrastructures achieve significant deployment cost reduction while always providing the required low latency (e.g., 20ms). In particular, a smart grid scenario was studied whereby existing low bandwidth networks need to be upgraded with high-bandwidth network links (e.g., optical fiber). The problem was formulated as an integer programming problem that minimizes the deployment cost for added high-bandwidth links. To solve it, the proposed algorithms take into account the characteristics of low latency applications together with topological characteristics to identify network locations where network capacities are insufficient. By doing so, the optimal network locations to install high-bandwidth links can be determined. In particular, the resulting infrastructure of one algorithm based on network calculus has guaranteed worst-case end-to-end latencies for deterministic workloads. The solution achieves 80% of deployment cost reduction compared to conventional approaches under a large set of real power grid topologies.

In Chapter 4, the cost-efficient design of dynamic resource management for low latency applications was investigated. Specifically, a fully NFV-enabled network was adopted to support MECs whereby end services can be instantiated at any network locations. The proposed dynamic resource management approach consists of a fast heuristic-based incremental allocation algorithm and an approximation-based reoptimization algorithm. The two algorithms jointly optimize end-to-end routing and computing resource allocation at end services, aimed at always exploiting the minimum amount of resources to achieve the required low latency. It was shown that the proposed approach achieves 33% operational cost reduction compared to current practices. Further, the in-depth impact factor analysis shows that the approximation algorithm achieves cost efficiency within 20% of the lower bound of the optimal solution, regardless of network size, services' latency requirements and MEC server capacities. Meanwhile, it was mathematically proved that given user demands, the proposed approximation algorithm results in no more than $ln(N)$ times of optimal operational cost in polynomial time, where N is the largest number of APs that are located within the latency constraint of the hosted application.

In Chapter 5, the cost-efficient design of fault-tolerant resource management was investigated, which considers the resource allocation of both active and back-up resources. In particular, the different stateful VNFs in a service function chaining are considered in this problem, whereby VNF states need to be transferred to the back-up VNFs upon failures. A fast heuristic was proposed, which jointly computes the placement of active and stand-by instances of stateful VNFs, the state transfer paths and the user request routings. For a special case of the problem without network bandwidth constraint, a bicriteria approximation algorithm with performance guarantees was designed. The evaluation results suggest that the proposed algorithms significantly improve the request admission rate while reducing DCs' operational cost.

In summary, the contribution of this thesis is the development of a novel cost-efficient resource allocation framework for the deployment of low-cost low latency communications. Specifically, the proposed framework solved the low latency communication problem from three different perspectives, addressing 1) the cost-efficient design of low latency communication infrastructures; 2) the cost-efficient design of dynamic resource management for low latency applications; and 3) the cost-efficient design of fault-tolerant resource management.

6.2 Future Research Directions

Robust and stochastic resource management: The current dynamic resource management approach presented in Chapter 4 minimizes the operational cost of the entire network at every time slot under the assumption that there is no state transfer cost. However, this is valid only for stateless low latency applications. To complement the study of dynamic resource management with stateful applications, we aim to further investigate stochastic resource management for stateful low latency applications. For such scenarios, changing the user-to-VM assignment incurs migration costs. Therefore, minimizing operational cost at each time slot may constantly incur significant migration cost. The idea of stochastic resource management is to minimize the average operational cost over a time period (rather than optimizing at each time slot) by taking the probability of having network dynamics such as user mobility over this period.

Neural network placement: In this thesis, the computing resource allocation for AR applications was studied in Chapter 4. However, this type of applications (e.g., face recognition and voice recognition) require a pre-trained neural network to identify objects, which will be placed in MECs. Then, users offload their computation tasks to MECs to exploit the corresponding neural network for identification. As a matter of fact, the trained models would consume a significant amount of data storage space, and pose challenges in storing them in edge systems. In general, the more accurate the neural network is, the larger space the trained model would consume. However, low latency networks such as edge networks cannot accommodate such neural networks in each MEC due to capacity limitations. As such, different neural networks with different accuracy and size will need to be intelligently placed into MECs. This necessitates research in both neural network training methods and placement algorithms.

Low latency big data analysis: This thesis has focused on achieving low latency communications and low latency service processing. However, the low latency data analysis has not been studied. In smart city applications, critical infrastructures need to have real-time monitoring to guarantee its operation, which requires fast data analysis to predict potential failures. Failing to perform data analysis in time could significantly affect the prediction outcome and the reliability of the infrastructures. To this end, the big data (monitoring data) placement for low latency data analysis needs to be studied in the future. In particular, we will study the adaptation of Hadoop (MapReduce) in a resource-constrained edge computing system. This consists in optimally selecting Map and Reduce nodes (i.e., where to store data) among edge nodes, so that the inter-node data transmission time can be reduced to lower the final data analysis time.

Bibliography

[1] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. Overlay: Practical mobile augmented reality. In *ACM Mobisys*, pages 331–344, 2015.

[2] Yue Cao, Houbing Song, Omprakash Kaiwartya, Bingpeng Zhou, Yuan Zhuang, Yang Cao, and Xu Zhang. Mobile edge computing for big-data-enabled electric vehicle charging. *IEEE Communication Magazine*, 56(3):150–156, 2018.

[3] Meryem Simsek, Adnan Aijaz, Mischa Dohler, Joachim Sachs, and Gerhard Fettweis. 5g-enabled tactile internet. *IEEE Journal on Selected Areas in Communications*, 34(3):460–473, 2016.

[4] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 18(3):1617–1655, 2016.

[5] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4), 2009.

[6] Mark Handley. Why the internet only just works. *BT Technology Journal*, 24(3):119–129, 2006.

[7] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[8] NFV ETSI. *NFV white paper 1.* https://portal.etsi.org/NFV/, 2014.

[9] Milica Pejanovic DJurisic, Zhilbert Tafa, Goran Dimic, and Veljko Milutinovic. A survey of military applications of wireless sensor networks. In *IEEE Mediterranean Conference on Embedded Computing*, pages 196–199, 2012.

[10] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Elsevier, Future generation computer systems*, 29(7):1645–1660, 2013.

[11] Guilherme Galante and Luis Carlos E de Bona. A survey on cloud computing elasticity. In *IEEE Conference on Utility and Cloud Computing*, pages 263–270, 2012.

[12] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, 9, 2011.

[13] Zhenhua Liu, Iris Liu, Steven Low, and Adam Wierman. Pricing data center demand response. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):111–123, 2014.

[14] Niels Bouten, Jeroen Famaey, Rashid Mijumbi, Bram Naudts, Joan Serrat, Steven Latré, and Filip De Turck. Towards nfv-based multimedia delivery. In *IEEE IM*, pages 738–741, 2015.

[15] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Elsevier, Future generation computer systems*, 28(5):755–768, 2012.

[16] Yongxu Zhu, Gan Zheng, Kai-Kit Wong, Shi Jin, and Sangarapillai Lambotharan. Performance analysis of cache-enabled millimeter

wave small cell networks. *IEEE Transactions on Vehicular Technology, DOI:10.1109/TVT.2018.2797047*, 2018.

[17] Wenye Wang, Yi Xu, and Mohit Khanna. A survey on the communication architectures in smart grid. *Elsevier, Computer Networks*, 55(15):3604–3629, 2011.

[18] R.H. Khan and J.Y. Khan. Wide area PMU communication over a WiMAX network in the smart grid. *IEEE SmartGridComm*, pages 187–192, 2012.

[19] K.V. Katsaros, Binxu Yang, Wei Koong Chai, and G. Pavlou. Low latency communication infrastructure for synchrophasor applications in distribution networks. *IEEE SmartGridComm*, pages 392–397, 2014.

[20] Lei Yang, Jiannong Cao, Guanqing Liang, and Xu Han. Cost aware service placement and load dispatching in mobile cloud systems. *IEEE Transactions on Computers*, 65(5):1440–1452, 2016.

[21] Giuliano Andrea Pagani and M. Aiello. The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

[22] Bozidar Radunović and Jean-Yves Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking*, 15(5):1073–1083, 2007.

[23] Binxu Yang, Wei Koong Chai, George Pavlou, and Konstantinos V Katsaros. Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud. In *IEEE CloudNet*, pages 136–141, 2016.

[24] Binxu Yang, Wei Koong Chai, Zichuan Xu, Konstantinos V Katsaros, and George Pavlou. Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications. *IEEE Transactions on Network and Service Management, DOI:10.1109/TNSM.2018.2790081*, 2018.

[25] I-Hong Hou, Tao Zhao, Shiqiang Wang, and Kevin Chan. Asymptotically optimal algorithm for online reconfiguration of edge-clouds. In *ACM MobiHoc*, pages 291–300, 2016.

[26] Andrew S Tanenbaum et al. Computer networks, 4-th edition. *Prentice Hall*, 2003.

[27] Binxu Yang, Zichuan Xu, Wei Koong Chai, Weifa Liang, Daphné Tuncer, Alex Galis, and George Pavlou. Algorithms for fault-tolerant placement of stateful virtualized network functions. *IEEE ICC, to appear*, 2018.

[28] Myungmoon Lee, Jintae Yu, Yongbum Kim, Chul-Hee Kang, and Jinwoo Park. Design of hierarchical crossconnect wdm networks employing a two-stage multiplexing scheme of waveband and wavelength. *IEEE Journal on Selected Areas in Communications*, 20(1):166–171, 2002.

[29] Robert LiKamWa and Lin Zhong. Starfish: Efficient concurrency support for computer vision applications. In *ACM MobiSys*, pages 213–226, 2015.

[30] Kiryong Ha, Padmanabhan Pillai, Grace Lewis, Soumya Simanta, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *IEEE International Conference on Cloud Engineering*, pages 166–176, 2013.

[31] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *IEEE annual workshop on network and systems support for games*, page 2, 2012.

[32] IEEE. IEEE Standard for Synchrophasor Measurements for Power Systems. *IEEE Standard C37.118.1*, 2011.

[33] Konstantinos V. Katsaros, Wei Koong Chai, Ning Wang, George Pavlou, Herman Bontius, and Mario Paolone. Information-centric Networking for

Machine-to-Machine Data Delivery - A Case Study in Smart Grid Applications. *IEEE Network*, 28(3):58–64, 2014.

[34] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Elsevier, Computer Networks*, 54(5):862–876, 2010.

[35] Anton Beloglazov and Rajkumar Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379, 2013.

[36] MEC ETSI. Mobile-edge computing. *Introductory Technical White Paper, September*, 2014.

[37] Joel Halpern and Carlos Pignataro. Service function chaining (sfc) architecture. Technical report, 2015.

[38] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, 2014.

[39] Daphne Tuncer, Marinos Charalambides, Stuart Clayman, and George Pavlou. Flexible traffic splitting in openflow networks. *IEEE Transactions on Network and Service Management*, 13(3):407–420, 2016.

[40] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. Extending networking into the virtualization layer. In *ACM Hotnets*, 2009.

[41] Lili Qiu, Venkata N Padmanabhan, and Geoffrey M Voelker. On the placement of web server replicas. In *IEEE INFOCOM*, pages 1587–1596, 2001.

[42] J.Kaplan, W.Forrest, and N.Kindler. Revolutionizing data center energy effiency. In *McKinsey and Company, Techical Reports, 2008*.

[43] Ramaswami Sridharan. The capacitated plant location problem. *Elsevier, European Journal of Operational Research*, 87(2):203–213, 1995.

[44] John G Klincewicz. Hub location in backbone/tributary network design: A review. *Elsevier, Location Science*, 6(1):307–335, 1998.

[45] Stefan Chanas and Dorota Kuchta. A concept of the optimal solution of the transportation problem with fuzzy cost coefficients. *Elsevier, Fuzzy sets and Systems*, 82(3):299–305, 1996.

[46] Julia Chuzhoy and Joseph Naor. Covering problems with hard capacities. *SIAM Journal on Computing*, 36(2):498–515, 2006.

[47] John Richard Current and James Edward Storbeck. Capacitated covering models. *Environment and planning B: planning and Design*, 15(2):153–163, 1988.

[48] Tom Leighton and Satish Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *IEEE Symposium on Foundations of Computer Science*, pages 422–431, 1988.

[49] Moses Charikar, Yonatan Naamad, Jennifer Rexford, and K Zou. Multi-commodity flow with in-network processing. *Manuscript, www. cs. princeton. edu/˜jrex/papers/mopt14. pdf*, 2014.

[50] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[51] John E Beasley. Lagrangean heuristics for location problems. *Elsevier, European Journal of Operational Research*, 65(3):383–399.

[52] Ricardo Lima. Ibm ilog cplex-what is inside of the box? In *EWO Seminar. Pittsburgh: Carnegie Mellon University*, 2010.

[53] Jian Li, Truong Khoa Phan, W Chai, Daphne Tuncer, George Pavlou, David Griffin, and Miguel Rio. Dr-cache: Distributed resilient caching with latency guarantees. In *IEEE INFOCOM, to appear*, 2018.

[54] IBM ILOG CPLEX. V12. 1: User manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[55] Amir Hossein Gandomi, Xin-She Yang, Siamak Talatahari, and Amir Hossein Alavi. Metaheuristic algorithms in modeling and optimization. In *Elsevier, Metaheuristic applications in structures and infrastructures*, pages 1–24. 2013.

[56] Ta Nguyen Binh Duong, Xiaorong Li, Rick Siow Mong Goh, Xueyan Tang, and Wentong Cai. Qos-aware revenue-cost optimization for latency-sensitive services in iaas clouds. In *IEEE Symposium on Distributed Simulation and Real Time Applications*, pages 11–18, 2012.

[57] Di Niu, Hong Xu, Baochun Li, and Shuqiao Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *IEEE INFOCOM*, pages 460–468, 2012.

[58] Hanan Luss. Operations research and capacity expansion problems: A survey. *Operations research*, 30(5):907–947, 1982.

[59] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[60] Anantaram Balakrishnan, TL Magnanti, A Shulman, and RT Wong. Models for planning capacity expansion in local access telecommunication networks. *Springer, Annals of Operations Research*, 33(4):237–284, 1991.

[61] Wenpeng Luan, Duncan Sharp, and Sol Lancashire. Smart grid communication network capacity planning for power utilities. In *IEEE PES Conference on Transmission and Distribution*, pages 1–4, 2010.

[62] Benazir Fateh, Manimaran Govindarasu, and Venkataramana Ajjarapu. Wireless network design for transmission line monitoring in smart grid. *IEEE Transactions on Smart Grid*, 4(2):1076–1086, 2013.

[63] Cedric F Lam, Hong Liu, Bikash Koley, Xiaoxue Zhao, Valey Kamalov, and Vijay Gill. Fiber optic communication technologies: What's needed for datacenter network operations. *IEEE Communications Magazine*, 48(7), 2010.

[64] Alberto Ceselli, Marco Premoli, and Stefano Secci. Cloudlet network design optimization. In *IFIP/IEEE Networking*, 2015.

[65] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[66] Mina Sedaghat, Francisco Hernandez-Rodriguez, and Erik Elmroth. A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *ACM Cloud and Autonomic Computing Conference*, page 6, 2013.

[67] Mansoor Alicherry and TV Lakshman. Network aware resource allocation in distributed clouds. In *IEEE INFOCOM*, pages 963–971, 2012.

[68] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *ACM Conference on High Performance Computing, Networking, Storage and Analysis*, page 49, 2011.

[69] Meng Wang, Xiaoqiao Meng, and Li Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *IEEE INFOCOM*, pages 71–75, 2011.

[70] Ming Mao, Jie Li, and Marty Humphrey. Cloud auto-scaling with deadline and budget constraints. In *IEEE/ACM Grid Computing*, pages 41–48, 2010.

[71] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on parallel and distributed systems*, 24(6):1107–1117, 2013.

[72] Jing Xu and Jose AB Fortes. Multi-objective virtual machine placement in virtualized data center environments. In *IEEE GreenCom*, pages 179–188, 2010.

[73] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *IEEE CLOUD*, pages 500–507, 2011.

[74] Sameep Mehta and Anindya Neogi. Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers. In *IEEE NOMS*, pages 363–370, 2008.

[75] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. A cost-aware elasticity provisioning system for the cloud. In *IEEE ICDCS*, pages 559–570, 2011.

[76] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. Capacitated cloudlet placements in wireless metropolitan area networks. In *IEEE LCN*, pages 570–578, 2015.

[77] Raul Landa, Marinos Charalambides, Richard G Clegg, David Griffin, and Miguel Rio. Self-tuning service provisioning for decentralised cloud applications. *IEEE Transactions on Network and Service Management*, 13:197–211, 2016.

[78] A Ascigil, Truong Khoa Phan, V Sourlas, I Psaras, and G Pavlou. On uncoordinated service placement in edge clouds. In *IEEE CloudCom*, 2017.

[79] Daphne Tuncer, Vasilis Sourlas, Marinos Charalambides, Maxim Claeys, Jeroen Famaey, George Pavlou, and Filip De Turck. Scalable cache management for isp-operated content delivery services. *IEEE Journal on Selected Areas in Communications*, 34(8):2063–2076, 2016.

[80] Mike Jia, Jiannong Cao, and Weifa Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, 2015.

[81] Dinh Thai Hoang, Dusit Niyato, and Ping Wang. Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In *IEEE WCNC*, pages 3145–3149, 2012.

[82] Mike Jia, Weifa Liang, Zichuan Xu, Meitian Huang, and Yu Ma. Qos-aware cloudlet load balancing in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing, DOI: 10.1109/TCC.2017.2786738*, 2018.

[83] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. On the effect of forwarding table size on sdn network utilization. In *IEEE INFOCOM*, pages 1734–1742, 2014.

[84] Meitian Huang, Weifa Liang, Zichuan Xu, Wenzheng Xu, Song Guo, and Yinlong Xu. Dynamic routing for network throughput maximization in software-defined networks. In *IEEE INFOCOM*, pages 1–9, 2016.

[85] Argyrios G. Tasiopoulos, Sameer G Kullkarni, Mayutan Arumaithurai, and Ioannis Psaras. Drench: A semi-distributed resource management framework for nfv based service function chaining. In *IFIP/IEEE Networking 2017*.

[86] Sevil Mehraghdam, Matthias Keller, and Holger Karl. Specifying and placing chains of virtual network functions. In *IEEE CloudNet*, pages 7–13, 2014.

[87] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. Near optimal placement of virtual network functions. In *IEEE INFOCOM 2015*, pages 1346–1354.

[88] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. Virtual network functions placement and routing optimization. In *IEEE CloudNet*, pages 171–177, 2015.

[89] Tarik Taleb, Adlen Ksentini, and Bruno Sericola. On service resilience in cloud-native 5g mobile systems. *IEEE Journal on Selected Areas in Communications*, 34(3):483–496, 2016.

[90] Shriram Rajagopalan, Dan Williams, and Hani Jamjoom. Pico replication: A high availability framework for middleboxes. In *ACM Proceedings of SoCC*, 2013.

[91] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, and Andrew Warfield. Remus: High availability via asynchronous virtual machine replication. In *USENIX NSDI*, pages 161–174, 2008.

[92] YaoZu Dong, Wei Ye, YunHong Jiang, Ian Pratt, ShiQing Ma, Jian Li, and HaiBing Guan. Colo: Coarse-grained lock-stepping virtual machines for non-stop service. In *ACM Proceedings of SoCC*, page 3, 2013.

[93] Yossi Kanizo, Ori Rottenstreich, Itai Segall, and Jose Yallouz. Optimizing virtual backup allocation for middleboxes. *IEEE/ACM Transactions on Networking*, 25(5):2759–2772, 2017.

[94] Konstantinos V. Katsaros, Wei Koong Chai, Barbara Vieira, and George Pavlou. Supporting smart electric vehicle charging with information-centric networking. *IEEE Conference Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pages 174–179, 2014.

[95] Yue Cao, Ning Wang, George Kamel, and Young-Jin Kim. An electric vehicle charging management scheme based on publish/subscribe communication framework. *IEEE Systems Journal*, PP(99):1–14, 2015.

[96] D.M. Laverty, D.J. Morrow, R.J. Best, and P.A. Crossley. Differential ROCOF relay for Loss-of-Mains protection of Renewable Generation using phasor measurement over Internet Protocol. *Integration of Wide-Scale Renewable Resources Into the Power Delivery System, CIGRE/IEEE PES Joint Symposium*, pages 1–7, 2009.

[97] P.T. Myrda and K. Koellner. NASPInet - The Internet for Synchrophasors. *Hawaii International Conference on System Sciences*, pages 1–6, 2010.

[98] Mario Paolone, Marco Pignati, Paolo Romano, Stela Sarri, Lorenzo Zanni, and Rachid Cherkaoui. A Hardware-in-the-Loop Test Platform for the Real-Time State Estimation of Active Distribution Networks using Phasor Measurement Units. *Proceedings of Cigré SC6 Colloquium*, 2014.

[99] A.G. Phadke and J.S. Thorp. Communication needs for wide area measurement applications. *5th International Conference on Critical Infrastructure*, pages 1–7, 2010.

[100] Wei Koong Chai, Ning Wang, Konstantinos V Katsaros, George Kamel, George Pavlou, Stijn Melis, Michael Hoefling, Bárbara Vieira, Paolo Romano, Styliani Sarri, et al. An information-centric communication infrastructure for real-time state estimation of active distribution networks. *IEEE Transactions on Smart Grid*, 6(4):2134–2146, 2015.

[101] M. Chenine, Kun Zhu, and L. Nordstrom. Survey on priorities and communication requirements for PMU-based applications in the Nordic Region. *IEEE PowerTech*, pages 1–8, 2009.

[102] Yongxu Zhu, Gan Zheng, Lifeng Wang, Kai-Kit Wong, and Liqiang Zhao. Content placement in cache-enabled sub-6 ghz and millimeter-wave multi-antenna dense small cell networks. *IEEE Transactions on Wireless Communications, DOI:10.1109/TWC.2018.2794368*, 2018.

[103] Peng Cheng, Li Wang, Bin Zhen, and Shihua Wang. Feasibility study of applying LTE to Smart Grid. *IEEE Smart Grid Modeling and Simulation*, pages 108–113, 2011.

[104] J. Brown and J.Y. Khan. Performance comparison of LTE FDD and TDD based Smart Grid communications networks for uplink biased traffic. *IEEE SmartGridComm*, pages 276–281, 2012.

[105] Mark S. Daskin. What you should know about location modeling. *Naval Research Logistics*, 55(4):283–294, 2008.

[106] Richard L. Francis, Timothy J. Lowe, and H. Donald Ratliff. Distance Constraints for Tree Network Multifacility Location Problems. *Operations Research*, 26(4):570–596, 1978.

[107] Giuliano Andrea Pagani and M. Aiello. Towards Decentralization: A Topological Investigation of the Medium and Low Voltage Grids. *IEEE Transactions on Smart Grid*, 2(3):538–547, 2011.

[108] K. Christakou, J. LeBoudec, M. Paolone, and D.-C. Tomozei. Efficient Computation of Sensitivity Coefficients of Node Voltages and Line Currents in Unbalanced Radial Electrical Distribution Networks. *IEEE Transactions on Smart Grid*, 4(2):741–750, 2013.

[109] IEEE Guide for Design, Operation, and Integration of Distributed Resource Island Systems with Electric Power Systems. *IEEE Std 1547.4-2011*, pages 1–54, 2011.

[110] Qiao Li, Tao Cui, Yang Weng, R. Negi, F. Franchetti, and M.D. Ilic. An Information-Theoretic Approach to PMU Placement in Electric Power Systems. *IEEE Transactions on Smart Grid*, 4(1):446–456, 2013.

[111] IEEE. IEEE Standard for Synchrophasor Data Transfer for Power Systems. *IEEE Standard C37.118.2*, 2011.

[112] S. Galli, A. Scaglione, and Zhifang Wang. Power Line Communications and the Smart Grid. *IEEE SmartGridComm*, pages 303–308, 2010.

[113] R. Ramaswamy, Ning Weng, and T. Wolf. Characterizing network processing delay. 3:1629–1634, 2004.

[114] I. Chlamtac, A. Farago, Hongbiao Zhang, and A. Fumagalli. A deterministic approach to the end-to-end analysis of packet flows in connection-oriented networks. *IEEE/ACM Transactions on Networking*, 6(4):422–431, 1998.

[115] J.-Y. Le Boudec and G. Hebuterne. Comments on “a deterministic approach to the end-to-end analysis of packet flows in connection oriented networks”. *IEEE/ACM Transactions on Networking*, 8(1):121–124, 2000.

[116] Hongbiao Zhang. A note on deterministic end-to-end delay analysis in connection oriented networks. 2:1223–1227, 1999.

[117] Rashid Mijumbi, Joan Serrat, Juan-luis Gorricho, Steven Latre, Marinos Charalambides, and Diego Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communication Magazine*, 54(1):98–105, 2016.

[118] Argyrios G Tasiopoulos, Ray Atarashi, Ioannis Psaras, and George Pavlou. On the bitrate adaptation of shared media experience services. In *ACM SIGCOMM Workshop on QoE-based Analysis and Management of Data Communication Networks*, pages 25–30, 2017.

[119] Kiryong Ha, Padmanabhan Pillai, Grace Lewis, Soumya Simanta, Sarah Clinch, Nikki Davies, and Mahadev Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *IEEE Conference on IC2E*, pages 166–176, 2013.

[120] Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65:3702–3712, 2016.

[121] Joao Soares, Carlos Goncalves, Bruno Parreira, Paulo Tavares, Jorge Carapinha, Joao Paulo Barraca, Rui L Aguiar, and Susana Sargento. Toward a telco cloud environment for service functions. *IEEE Communication Magazine*, 53(2):98–106, 2015.

[122] Tarik Taleb and Adlen Ksentini. Follow me cloud: interworking federated clouds and distributed mobile networks. *IEEE Network*, 27(5):12–19, 2013.

[123] Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin Chan, and Kin K Leung. Dynamic service migration in mobile edge-clouds. In *IFIP/IEEE Networking*, pages 1–9, 2015.

[124] Yingwu Zhu and Yiming Hu. Efficient, proximity-aware load balancing for dht-based p2p systems. *IEEE Transactions on Parallel and Distributed Systems*, 16(4):349–361, 2005.

[125] Hao Yin, Xu Zhang, Hongqiang Liu, Yan Luo, Chen Tian, Shuoyao Zhao, and Feng Li. Edge provisioning with flexible server placement. *IEEE Transactions on Parallel and Distributed Systems, DOI:10.1109/TPDS.2016.2604803*.

[126] Anil Madhavapeddy, Thomas Leonard, Magnus Skjegstad, Thomas Gazagnaire, David Sheets, Dave Scott, Richard Mortier, Amir Chaudhry, Balraj Singh, Jon Ludlam, et al. Jitsu: Just-in-time summoning of unikernels. In *USENIX NSDI*, pages 559–573, 2015.

[127] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. Clickos and the art of network function virtualization. In *USENIX*, pages 459–473, 2014.

[128] Liang Tong, Yong Li, and Wei Gao. A hierarchical edge cloud architecture for mobile computing. In *IEEE INFOCOM*, pages 1–9, 2016.

[129] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. A parsimonious model of mobile partitioned networks with clustering. In *IEEE Communication Systems and Networks and Workshops*, pages 1–10, 2009.

[130] Stavros G Kolliopoulos and Clifford Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31(3):919–946, 2001.

[131] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

[132] Andras Varga. *OMNeT++ Simulator Home Page*. http://www.omnetpp.org.

[133] Dominik Klein and Michael Jarschel. An openflow extension for the omnet++ inet framework. In *ICST Conference on Simulation Tools and Techniques*, 2013.

[134] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[135] Qiufen Xia, Weifa Liang, and Wenzheng Xu. Throughput maximization for online request admissions in mobile cloudlets. In *IEEE LCN*, pages 589–596, 2013.

[136] Debessay Fesehaye, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. Impact of cloudlets on interactive mobile cloud applications. In *IEEE Conference on Enterprise Distributed Object Computing*, pages 123–132, 2012.

[137] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. Efficient algorithms for capacitated cloudlet placements. *IEEE Transactions on Parallel and Distributed Systems*, 27:2866–2880, 2016.

[138] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *IEEE CNSM*, pages 9–16, 2010.

[139] Kenneth L Calvert, Matthew B Doar, and Ellen W Zegura. Modeling internet topology. *IEEE Communication Magazine*, 35(6):160–163, 1997.

[140] Zichuan Xu, Weifa Liang, Alex Galis, and Yu Ma. Throughput maximization and resource optimization in nfv-enabled networks. In *IEEE ICC*, 2017.

[141] Stuart Clayman, Elisa Maini, Alex Galis, Antonio Manzalini, and Nicola Mazzocca. The dynamic placement of virtual network functions. In *IEEE NOMS*, 2014.

[142] Rahul Potharaju and Navendu Jain. Demystifying the dark side of the middle: a field study of middlebox failures in datacenters. In *ACM IMC*, pages 9–22, 2013.

[143] Wei Koong Chai, Vaios Kyritsis, Konstantinos V Katsaros, and George Pavlou. Resilience of interdependent communication and power distribution networks against cascading failures. In *IFIP/IEEE Networking*, pages 37–45, 2016.

[144] Francisco Carpio, Samia Dhahri, and Admela Jukan. Vnf placement with replication for loac balancing in nfv networks. In *IEEE ICC*, 2017.

[145] Babu Kothandaraman, Manxing Du, and Pontus Sköldström. Centrally controlled distributed vnf state management. In *ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, pages 37–42, 2015.

[146] Junaid Khalid, Aaron Gember-Jacobson, Roney Michael, Anubhavnidhi Abhashkumar, and Aditya Akella. Paving the way for nfv: Simplifying middlebox modifications using statealyzr. In *USENIX NSDI*, pages 239–253, 2016.

[147] Meitian Huang, Weifa Liang, Zichuan Xu, Mike Jia, and Song Guo. Throughput maximization in software-defined networks with consolidated middleboxes. In *IEEE LCN*, pages 298–306, 2016.

[148] Hernani D Chantre and Nelson LS da Fonseca. Redundant placement of virtualized network functions for lte evolved multimedia broadcast multicast services. In *IEEE ICC*, 2017.

[149] Huawei Huang, Song Guo, Jinsong Wu, and Jie Li. Service chaining for hybrid network function. *IEEE Transactions on Cloud Computing, DOI:10.1109/TCC.2017.2721401*, 2017.

[150] Stavros G Kolliopoulos and Clifford Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31(3):919–946, 2001.

[151] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[152] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[153] Inc Amazon Web Services. *Amazon ec2 instance configuration*. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html.

[154] Yang Li, Linh Thi Xuan Phan, and Boon Thau Loo. Network functions virtualization with soft real-time guarantees. In *IEEE INFOCOM*, pages 1–9, 2016.

[155] Microsoft. *Plan network requirements for Skype for business*. https://technet.microsoft.com/en-us/library/gg425841.aspx.