Energy Efficient Composable Data Centres

Opeyemi Oluwaseyi Ajibola

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

The University of Leeds
School of Electronic and Electrical Engineering

March, 2021

Intellectual Property Statement

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Chapter 4 is based on the work from:

O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “On Energy Efficiency of Networks for Composable Datacentre Infrastructures,” in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, 2018, pp. 1–5.

Prof Elmirghani, the supervisor, suggested the study of energy efficiency of networks in composable data centre infrastructures. The co-supervisor, Dr El-Gorashi, worked with the student to develop the MILP model, analyse the results and prepare the paper, Prof. Elmirghani checked the MILP model and the results. The PhD student developed the model, obtained and analysed results and wrote the paper.

And:

O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Energy Efficient Placement of Workloads in Composable Data Center Networks,” IEEE/OSA Journal of Lightwave Technology, 3 March 2021, DOI: 10.1109/JLT.2021.3063325.

Prof Elmirghani, the supervisor, suggested the study of energy efficiency of networks in composable datacentre infrastructures. The co-supervisor, Dr El-Gorashi, worked with the student to develop the MILP model and heuristic, analyse the results and prepare the paper, Prof. Elmirghani checked the MILP model and the results. The PhD student developed the model and heuristic, obtained and analysed results and wrote the paper.

Chapter 5 is based on the work from:

O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “A Network Topology for Composable Infrastructures,” in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1–4.

Prof Elmirghani, the supervisor, suggested the investigation and the design of an energy efficient network topology for composable datacentre infrastructures. The co-supervisor, Dr El-Gorashi, worked with the student to develop the MILP model, analyse the results and prepare the paper, Prof. Elmirghani checked the MILP model and the results. The PhD student developed the model, obtained and analysed results and wrote the paper.

And:

O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Network Topologies for Composable Data Centres,” submitted for publication in IEEE/OSA Journal of Lightwave Technology.

Prof Elmirghani, the supervisor, suggested the investigation and the design of an energy efficient network topology for composable datacentre infrastructures. The co-supervisor, Dr El-Gorashi, worked with the student to develop the MILP model, analyse the results and prepare the paper, Prof. Elmirghani checked the MILP model and the results. The PhD student developed the model, obtained and analysed results and wrote the paper.

Chapter 6 is based on the work from:

O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Disaggregation for Improved Efficiency in Fog Computing Era,” in *2019 21th International Conference on Transparent Optical Networks (ICTON)*, 2019, pp. 1–7.

Prof Elmirghani, the supervisor, suggested the application of the server disaggregation concept in the fog computing tier. The co-supervisor, Dr El-Gorashi, worked with the student to develop the MILP model, analyse the results and prepare the paper, Prof. Elmirghani checked the MILP model and the results. The PhD student developed the MILP model, obtained and analysed results and wrote the paper.

And:

O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Disaggregation for Energy Efficient Fog in Future 6G Networks” submitted for publication in IEEE Internet of Things Journal.

Prof Elmirghani, the supervisor, suggested the application of the server disaggregation concept in the fog computing tier. The co-supervisor, Dr El-Gorashi, worked with the student to develop the MILP model and heuristic, analyse the results and prepare the paper, Prof. Elmirghani checked the MILP model and the results. The PhD student developed the MILP model and heuristic, obtained and analysed results and wrote the paper.

Acknowledgements

Firstly, I thank God Almighty, who gave me wisdom and has brought me to this expected end in good health. I would like to express my gratitude to my supervisor Prof. Jaafar Elmirghani for the support during my Ph.D. and for his patience, guidance, and leadership all-through my studies. His insightful advices were invaluable toward accomplishing the goals of this work.

I would like to thank my second supervisor Dr. Taisir Elgorashi for the unending support, encouragement, and the insightful discussions I had with her during my Ph.D.

I would like to acknowledge the Petroleum Technology Trust Fund (Nigeria) for funding my Ph.D. study and for their support. I would like to thank Federal University Oye-Ekiti for the leave granted for my Ph.D. study.

I wish to express my gratitude to my entire family and in-laws for their unyielding support and encouragement. I acknowledge my Father Chief Simeon Sule Ajibola for believing in me and for the unending support all-through my life and during my Ph.D. study.

I thank my friends for encouraging words and support over the last four years.

I thank my colleagues in Communication Systems and Networks group for useful discussions and the support they provided for the last four years.

Finally, I wish to express my love and deepest gratitude to my wife and daughter, Oluwabukola and Oreoluwa, for their support, patience and understanding during my Ph.D. study.

Abstract

There is a proliferation of the number of operational data centres (DCs) across the globe to meet present and future demands for on-demand computational offerings. In spite of the many efforts to improve the utilisation and power efficiency of traditional DCs, results achieved remain sub-optimal. This is primarily because of the rigid utilisation boundaries of traditional server architecture. Disaggregation of server resource components and dynamic orchestration of such resources over a suitable network has been proposed to improve efficiency of next generation composable DCs. This thesis conducts a study on the best setup for such composable DC infrastructure to achieve optimal energy efficiency.

First, we formulate a mixed integer linear programming (MILP) model to investigate the optimal scale and scope of resource disaggregation for energy efficient composable DCs. Concurrently, we also investigate the most suitable network for optimal energy efficiency. By placing CPU and memory intensive workloads energy efficiently in different composable DCs, we found that implementing logical disaggregation at rack-scale in composable DCs that adopt all-optical network enables optimal energy efficiency. Physical resource disaggregation of traditional DC servers at rack-scale leads to up to 8% and 20% savings in overall power consumption when CPU intensive and memory intensive workloads are provisioned respectively. We also found that adoption of micro-service architecture in conjunction with the logical disaggregation and rack-scale resource disaggregation can further improve efficiency in composable DCs. A combination of disaggregation and micro-services enabled optimal resources utilisation and energy efficiencies. Thus, relative to the traditional DC up to 23% reduction in the total power consumption is achieved by combining both approaches.

Secondly, we describe two variants of a practical and scalable network for composable DC that leverages optical technologies and techniques. Additionally, we formulate a MILP model to evaluate the performance of the novel network in rack-scale composable DCs that implement different forms of disaggregation. The electrical-optical variant of the novel topology achieves similar performance as a reference network while utilising fewer transceivers per compute node. The targeted adoption of optical technologies by both variants of the proposed network achieves greater (4 - 5 times greater) utilisation of available network throughput than the reference network which implemented a generic design.

Furthermore, we also formulate a MILP model and develop a comparable heuristic to study the benefits of adopting server disaggregation in the fog computing tier. We evaluate the energy efficient placement of interactive apps in a future fog 6G network

in our study. Relative to the present practice of deploying traditional servers in the fog computing layer, adoption of disaggregated servers reduces total fog computing power consumption by up to 18% when a network with low delay penalty is considered.

Finally, we recommend that logical disaggregation and rack-scale disaggregation should be implemented in composable DCs that desire energy efficiency. This is because of the advantages and flexibility that both approaches jointly offer as reported in this thesis. We also recommend the targeted use of optical network technologies and techniques. Relative to a general-purposed design, this provides a more efficient approach to mitigate network challenges of composable DCs. Furthermore, these recommendations should be extended to the fog computing tier and edge of future networks to enable greater energy efficiency of the cloud-of-things architecture.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

5G	Fifth Generation
6G	Sixth Generation
AI	Artificial Intelligence
AMPL	A Mathematical Programming Language
AoD	Architecture on Demand
AOPD-DCN	All Optical Programmable Disaggregated Data Centre Network
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
AWG	Arrayed Waveguide Grating
AWGR	Arrayed Waveguide Grating Router
CAPEX	Capital Expenditure
CMOS	Complementary Metal Oxide Semiconductor
CO	Central Office
CPE	Consumer Premises Equipment
CPU	Central Processing Unit
CS	Cell Site
DC	Data Centre
DCN	Data Centre Network
DEMUX/DMX	De-multiplexer
DRAM	Dynamic Random-Access Memory
DS	Disaggregated Server
E-NetCoD	Electrical Network for Composable Data Centre
EO	Enterprise Office
EO-NetCoD	Electrical-Optical Network for Composable Data Centre
FaaS	Fog as a Service
FBG	Fibre Bragg Grating
FIaaS	Fog Infrastructure as a Service

FPGA	Field Programmable Gate Array
GB	Giga Byte
GbE	Giga bit Ethernet
Gbps	Giga bit per Second
GHz	Gigahertz
HA	High Availability
HCI	Hyper Converged Infrastructure
HDD	Hard Disk Drive
HEEP	Heuristic for Energy Efficient Placement
HEEDAP	Heuristic for Energy Efficient and Delay Aware Placement
IaaS	Infrastructure as a Service
IC	Integrated Circuit
ILP	Integer Linear Programming
IMCG	In-Memory Communication Group
IO	Input/Output
IoT	Internet of Things
IP	Internet Protocol
IRC	Idle Resource Capacity
MAN	Metropolitan Area Network
MILP	Mixed Integer Linear Programming
ML	Machine Learning
ms	Millisecond
MUX	Multiplexer
NCH	Node Controller Hub
NetCoD	Network for Composable Data Centre
NFV	Network Function Virtualisation
NIC	Network Interface Card
NG-PON2	Next Generation Passive Optical Network2
OCP	Open Compute Project

OCS	Optical Circuit Switched
OEO	Optical Electrical Optical
OLT	Optical Line Terminal
ONU	Optical Network Unit
OPEX	Operational Expenditure
OPS	Optical Packet Switched
OS	Operating System
OXC	Optical Cross Connect
PaaS	Platform as a Service
PON	Passive Optical Network
PUE	Power Usage Effectiveness
RAM	Random Access Memory
RDMA	Remote Direct Memory Access
RSD	Rack Scale Design
RTT	Round Trip Time
SaaS	Software as a Service
SAN	Storage Area Network
SCM	Storage Class Memory
SD	Software Defined
SDC	Software Defined Compute
SDDC	Software Defined Data Centre
SDH	Synchronous Digital Hierarchy
SDI	Software Defined Infrastructure
SDM	Space Division Multiplexing
SDN	Software Defined Network
SDS	Software Defined Storage
SIC	Switch Interface Card
SLA	Service Level Agreement
SOA	Semiconductor Optical Amplifier

SoC	System on Chip
SONET	Synchronous Optical Networking
TA	Traditional Application
TCO	Total Cost Owner
ToC	Top of Cluster
ToR	Top of Rack
TS	Traditional Server
VM	Virtual Machine
VNF	Virtual Network Function
WDM	Wavelength Division Multiplexing
WSS	Wavelength Selective Switch

Chapter 1 : Introduction

Datacentres (DCs) are critical infrastructures that provide platforms driving wide adoption of digital technologies. These indispensable infrastructures provide computing resources needed to run public internet-facing applications and private enterprise-critical applications alike. DCs provide environments that satisfy the requirements of cloud computing and data analytics applications. Such requirements include on-demand resource provisioning, multitenant isolation, parallel computation, and security. Examples of cloud computing and data analytics applications include web services, web-search, instant messaging and social media, distributed file systems, analytics, and content delivery applications. A typical DC comprise of compute, storage and network resources and peripheral systems. A traditional server is the basic compute unit in traditional DCs. Each server is a modular node with intrinsic CPU, memory, storage, and network resources. The number of severs in a traditional DC ranges from a few hundreds to tens or hundreds of thousands of servers. This number varies based on the purpose for which a DC is built.

Cloud computing became an integral part of the global society over the last two decades. This is because cloud computing enabled improved capital and operational efficiencies relative to the traditional distributed computing architecture. Adoption of cloud computing in its different service offerings such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) spans across a variety of personal, enterprise and public applications. In 2020, it is predicted that 83% of enterprise workloads will be in the cloud. Some of the key drivers of this trend include digital transformation, artificial intelligence and machine learning and Internet of Things (IoT) [1]. By 2022, it is estimated that the global spending on IoT will reach $1.2 trillion [2]. Furthermore, Cisco predicts that about 500 billion smart devices will be connected by 2030 [3]. In addition to these widely used cloud computing applications, strong emergence of other disruptive technologies is also expected to increase the variety of applications deployed in datacentres (DCs) [4] in the near future. Examples of such disruptive technologies include network function virtualisation, big data analytics and smart grids and smart cities. Consequentially, the number and size of DCs is expected to increase. The expected growth in the proliferation of DCs and DC applications has become an important motivation for increased flexibility, resource utilisation efficiency and energy efficiency. This is essential in DCs to attain desired

performance at low cost, energy efficiently and in an environment friendly and sustainable manner.

Furthermore, the growing uptake of the cloud computing and IoT paradigms is expected to enable a new range of applications at the edge of telecommunication networks. However, the quasi-distributed or semi-centralised architecture of the traditional cloud computing paradigm is a major inhibiting factor to the emergence of some of applications. This is because such application may require real-time or near-real-time computation. In addition, the volume, velocity, and variety of data generated by geo-distributed IoT-devices and end-users of these emerging Cloud-IoT applications combined with traditional network traffic is expected to overwhelm network infrastructures. This will also significantly increase the cost of owning and operating these networks and may increase their carbon footprint. Ultimately, the centralised cloud computing architecture will degrade the performance of some future applications due to increased network congestion. In other cases, the centralised cloud computing architecture will outrightly prevent the emergence of some applications which are infeasible under the centralised cloud computing architecture. Hence, the concept of edge/fog computing [5]–[7] has been proposed in recent times to address some of these challenges.

The fog computing paradigm extends cloud computing to the network edge to support emerging and future internet applications and services. This is enabled via the introduction of a new intermediate computation tier called the fog computing tier. The fog computing tier is located between the centralised cloud computing tier and geo-distributed IoT-devices and end-users of emerging applications at the network edge. The fog tier comprises of heterogeneous devices and nodes called fog nodes. Examples of fog nodes include edge routers, access points, specialised servers, and a range of endpoints such as connected vehicles, surveillance cameras and mobile phones. These heterogeneous fog nodes/devices adopt heterogeneous network infrastructures (both wireless and wired) for connectivity. The geo-distributed heterogeneous nodes may also be orchestrated collectively as a fog federation to enable a fog as a service (FaaS) or fog Infrastructure as a service (IaaS, FIaaS) business model [8]–[10]. The goals of the fog computing paradigm include: minimisation of response time for real-time and mission critical applications (such as vehicle to vehicle communication, intelligent processing/analytics, wireless sensors and actuation networks and game streaming) via in-situ computation [11], [12]; the reduction of cloud computing

destined workloads; and the reduction of total end to end traffic in networks [11].

In recent times, hyper-scale DC infrastructure providers such as Facebook and Microsoft have begun to explore the use of the server resource disaggregation concept. They aim to use server resource disaggregation as a tool to further improve on DC infrastructure overall efficiency [13]. Server resource disaggregation proposes the physical or logical separation of traditional server (TS) intrinsic resources into pools of homogeneous resources. Such resources are subsequently composed, decomposed, and recomposed on-demand over high bandwidth and low latency networks, to support applications. Hence, a composable DC or computing infrastructure is enabled. This concept addresses the limitations associated with computing infrastructures and DCs that employ traditional server architecture. Such limitations include poor resource modularity and lifecycle management, the need for purpose-built servers in computing clusters, high power consumption and capital expenditure resulting from inefficient resource utilisation [14]–[17]. Furthermore, the adoption of the composable infrastructure concept in the fog computing tier can also enable potential improvements in fog computing efficiency. Consequently, fog computing efficiency could approach efficiencies that are traditionally attributed to the cloud computing tier. In spite of the potential benefits attainable when resource disaggregation is implemented to achieve a composable DC, there are a few inherent implementation challenges and open issues that must be addressed.

For example, although different scales of resource disaggregation are possible in composable DCs, the optimal and practical scale of resource disaggregation in composable DC infrastructures remains an open issue. Furthermore, in spite of some preliminary works on the design of suitable network topologies to support resource disaggregation in composable DCs with little or no performance degradation, additional research is required. Such research is required to achieve practical networks to maximise the benefits enabled by resource disaggregation. In this thesis, we compare different scales and scopes of resource disaggregation in composable DC infrastructures via the formulation of a mixed integer linear programming (MILP) model. We also review electrical, optical and hybrid network topologies proposed for composable DC infrastructure in existing literature and evaluate the performance of selected prototype topologies. The concept of logical resource disaggregation in composable DC where resource utilisation boundaries are virtually relaxed is also explored. We also evaluate the impact

of using micro-services to form integrated workloads in both traditional and composable DCs over the deployment of monolithic workloads. A generic heuristic for energy efficient placement of workload resource demands into a rack-scale composable DC is also proposed in this thesis.

Furthermore, to unleash and to maximise the potential benefits of employing resource disaggregation in composable DCs, two variants of a high-capacity, simple and practical network for composable DC are proposed in this thesis. The proposed topologies leverage optical networking component, technologies, and techniques. Consequently, full mesh physical topology within each rack in the composable DC is achieved while using minimal number of interfaces and transceivers. A MILP model is formulated as proof of concept of the proposed topologies. The MILP model also aided comprehensive evaluations of the proposed network topology relative to a reference network topology in the literature. We also evaluate the overall energy efficiency of composable DCs that implement the novel network topologies by extending the MILP model to energy efficiently provision workload demands.

Additionally, although adoption of the disaggregated servers at the edge of the network seems intuitively practical, a comprehensive study is necessary to validate the proposition relative to the use of traditional servers. Furthermore, it is also important to investigate the impact of such proposition on existing network infrastructure and the performance of interactive fog applications. Therefore, in this thesis, a MILP model is formulated to place delay sensitive and insensitive fog applications in a metro-access fog computing tier. The metro-access fog computing tier replaces traditional servers with disaggregated servers. We also study the impact of user-distribution in a network of federated fog computing sites in the metro and access networks. Finally, we develop a fast and scalable policy (heuristic) which mimics the MILP model for practical deployment in large scenarios, and to verify the MILP.

1.1 Research Objectives

The primary research objectives presented in this thesis include:

1. To determine the optimal and practical scale and scope of resource disaggregation in composable DCs by formulating a MILP model that performs energy efficient placement of workloads over electrical, optical and hybrid network topologies.
2. To design a novel, simple, practical, and scalable network for composable DCs and to study the performance of the network by

formulating a MILP model that maximises network throughput and another MILP model that performs energy efficient routing and forwarding over the novel network.

3. To further evaluate the performance of the proposed network in composable DCs by formulating a MILP model that places workload resource demands energy efficiently in composable DCs that deploy the novel network.
4. To investigate the impact of adopting disaggregated servers in metro and access networks comprising of federated fog computing sites relative to the adoption of traditional servers by formulating a MILP model that performs energy efficient placement of delay sensitive and insensitive fog applications at the edge of the network.
5. To establish the factors that impact the energy efficient placement of delay sensitive and insensitive fog applications at the edge of the network and to determine optimal fog computing sites, at the edge of the network, for present and future fog applications.

1.2 Original Contributions

The main contributions of this thesis include:

1. The formulation of a MILP models that performs energy efficient placement of monolithic workloads in different types of composable DCs over electrical, hybrid and optical networks. The MILP model demonstrated that disaggregation at rack-scale in a composable DC that employs an optical network is sufficient to achieve optimal efficiency. The model was extended to consider the impact of replacing monolithic workloads in rack-scale composable DCs with comparable integrated workloads. The integrated workload is created concurrently by provisioning more modular related micro-services derived from decomposing a monolithic workload. We showed that up to 23% reduction in total DC power consumption can be achieved by deploying integrated workloads in a composable DC relative the deployment of monolithic workloads in a traditional DC.
2. A heuristic for energy efficient placement (HEEP) of workloads in rack-scale composable DCs that deploys an optical network was developed for real-time placement of workloads in composable DCs. Compared to the MILP, the HEEP algorithm achieved comparable power consumption.
3. An electrical and an electrical-optical variants of a novel, simple and practical network for composable DCs (NetCoD) are proposed in this thesis. As a proof of concept, a MILP formulation that models both proposed variants of NetCoD and another reference network topology from the literature is formulated. The performance of the networks were compared using the MILP formulation. Under similar loads and physical setup, it was observed that both variant of NetCoD achieved comparable performance as the reference network which does not scale well for practical DCs. The MILP was extended to study the

performance of all networks when energy efficient placement of workloads is required in a rack-scale composable DC. It was observed that Electrical-Optical-NetCoD (EO-NetCoD) achieved comparable performance and power consumption as the reference topology. On the other hand, the Electrical-NetCoD (E-NetCoD) incurred higher power consumption while achieving comparable performance as both EO-NetCoD and the reference network.

4. A MILP was formulated to study the impact of adopting the disaggregated servers in an energy efficient fog computing tier. The adoption of disaggregated servers achieved up to 18% reduction in total fog computing power consumption relative to the use of traditional servers in the fog computing layer. It was also observed that more instances of interactive fog applications are provisioned when a fog network is implemented in metro and access networks with high delay penalty. The proximity of metro central offices and radio cell sites to geo-distributed users of interactive fog applications (apps) made them suitable locations for provisioning moderately delay sensitive fog applications to enable optimal energy efficiency in the fog computing tier.
5. A heuristic for energy efficient and delay aware placement (HEEDAP) of workloads in metro-fog network was proposed. The HEEDAP algorithm achieved comparable performance and power savings relative to the MILP.

1.3 Related Publications

The following publications support the original contributions in this thesis:

Journals

1. O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Energy Efficient Placement of Workloads in Composable Data Center Networks,” IEEE/OSA Journal of Lightwave Technology, 3 March 2021, DOI: 10.1109/JLT.2021.3063325.
2. O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Disaggregation for Energy Efficient Fog in Future 6G Networks,” submitted for publication in IEEE Internet of Things Journal.
3. O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Network Topologies for Composable Data Centres,” submitted for publication in IEEE/OSA Journal of Lightwave Technology.

Conferences

4. O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “On Energy Efficiency of Networks for Composable Datacentre Infrastructures,” in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, 2018, pp. 1–5.
5. O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “Disaggregation for Improved Efficiency in Fog Computing Era,” in

2019 21th International Conference on Transparent Optical Networks (ICTON), 2019, pp. 1–7.
6. O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani, “A Network Topology for Composable Infrastructures,” in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1–4.

1.4 Outline of Thesis

Following the introduction in Chapter 1, the rest of this thesis is organised as follows:

Chapter 2 gives an overview of important components and concepts in this thesis. A review of DC infrastructure evolution along with software and infrastructure centric techniques used to enable greater efficiency in DCs are given. A brief review of WDM networks and enabling optical devices and architecture are also presented in Chapter 2. The Chapter also gives description of the virtual topology design problem in WDM networks, the MILP, non-linear programming and metaheuristic optimisation approaches.

Chapter 3 presents an overview of composable DC infrastructure along with a review of its enabling technologies and implementation challenges. We also review research on composable DC infrastructure from existing literature.

Using MILP, Chapter 4 conducts a study of energy efficient placement of different types of workloads in a range of composable DC infrastructure over selected electrical, hybrid and optical networks. MILP is also used to compare the adoption of integrated workloads, comprising of more modular related micro-services, to the adoption of monolithic workloads in both traditional and composable DCs. The chapter presents a heuristic for energy efficient placement (HEEP) of workloads in DCs and compares the performance of HEEP to the MILP model formulated.

In Chapter 5, the description of a novel, simple and practical network for composable DC is given. Using MILP, the chapter evaluates the performance of the novel network topology relative to an existing network from the literature. Furthermore, MILP is also used to perform energy efficient placement of workloads in a range of rack-scale composable DC infrastructures that deploy the proposed topology. The chapter also compares the performance of the novel network to that of the reference network topology from the literature after the energy efficient placement of workloads.

Chapter 6 investigates the benefits and impact of adopting disaggregated servers for greater energy efficiency in the fog computing era. In this chapter, a MILP model is formulated to optimally provision delay

sensitive and insensitive fog apps in a fog network. The fog network comprises of federated fog computing sites in the metro and access networks. The chapter also proposes a heuristic for energy efficient and delay aware placement (HEEDAP) of fog applications in a fog network at the edge of the network. Finally, the chapter compares the performance of the HEEDAP algorithm to that of the formulated MILP model.

Chapter 7 concludes this thesis. A summary of the main contributions in the thesis are given and future works are proposed.

Chapter 2 : Review of Data Centre Infrastructure and Wavelength Division Multiplexing (WDM) Networks

2.1 Introduction

In this chapter, an overview of important components and concepts in this thesis is presented. A review of data centre (DC) infrastructure evolution is given along with an overview of software and hardware techniques used to enable greater efficiency in DCs. We follow this with a brief overview of WDM networks and enabling optical devices and architectures. We also give a brief overview of the virtual topology design problem in WDM networks, which is considered in this thesis and is often formulated as a mixed integer linear programming (MILP) model. Finally, we give a short review of the MILP optimisation approach used widely in this thesis and we briefly review non-linear programming and metaheuristic optimisation methods.

2.2 Traditional Data Centre infrastructure

A data centre (DC) comprises of compute, storage and network resources and peripheral systems. A server is the basic compute unit in DCs. Each server is a modular node with intrinsic CPU, memory, storage, and network resources. In traditional DCs, servers were bespoke hardware designed to support a single instance of a monolithic application. Hence, one to one allocation of monolithic application instances to servers contributed to low (between 10-30%) utilisation of installed resource capacity. Up to 48 servers are placed in cabinet-like structures called racks, servers within a rack are inter-connected by an intra-rack communication network. Multiple co-located racks are connected by an inter-rack network to form a DC cluster. Peripheral systems including management and orchestration platforms, power and cooling systems are critical for daily operation and management of DCs.

A network topology in the DC connects several servers, storage devices and other resources together to support east-west traffic (within each DC) and north-south traffic (between the DC and users or other remote DCs). The legacy network topology deployed in traditional DC was a three-tier hierarchical topology comprising of access, aggregation and core layers [18] as shown in Figure 2.1. In the access layer, each rack accommodates multiple servers. Each rack has a Top of Rack (ToR) switch that connects to these servers via 1 Gbps or 2 Gbps links. The aggregation layer consists of aggregation switches interconnecting multiple ToR switches via 10 Gbps links.

Aggregation switches route traffic between different racks and from ToR switches to the core layer. The core layer consists of core routers which are intermediate routers between the DC and the Internet. The aggregation switches are connected to the core routers via 10 Gbps or 100 Gbps links. In the three-tier hierarchical topology, there are no physical links between switches in the same layer, links only exists between switches of adjacent layers. The traditional DC network is limited because oversubscribed links in higher layers of the network topology become bottlenecks when the volume of traffic in the lower layer links increases. Traditional DC networks provide poor support for on-demand horizontal scaling of servers. Traditional DC networks also have a single point of failure as a failure in a higher-layer switch or link can affect several racks and servers [19].

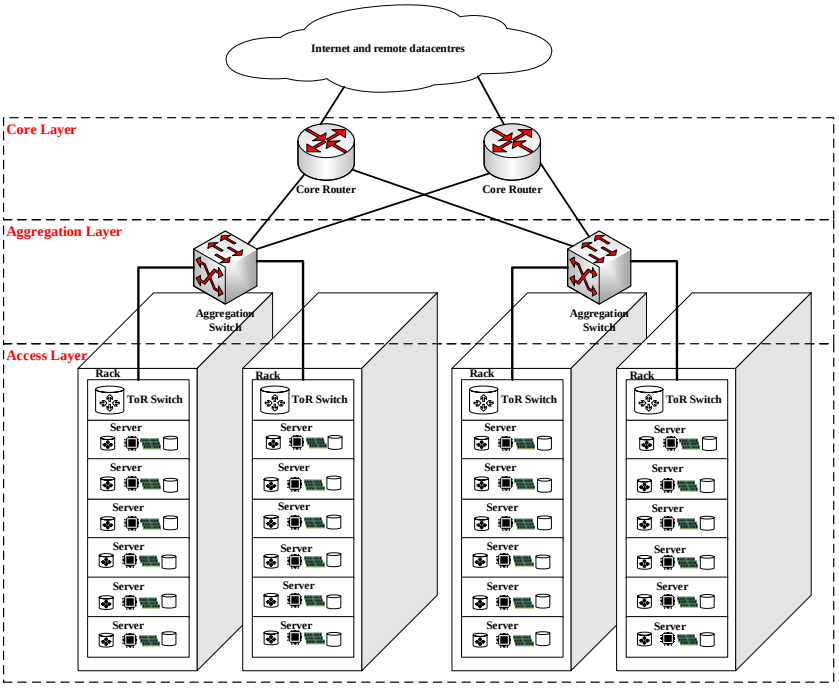


Figure 2.1: Traditional datacentre infrastructure

DCs have been classified into university campus, private enterprise and cloud categories [18], [20]. This classification is based on the mix of applications in the DC and the organisation that owns and operates the infrastructure. On the one hand, university campus and private enterprise DCs are on-premises infrastructures with a few hundreds of servers. On the other hand, cloud DCs (operated by cloud providers such as Amazon, Google, Microsoft and Facebook) usually comprise of tens or hundreds of thousands of servers. These servers are often distributed across the globe into distributed cloud DCs to ensure that user's quality of service and regulatory requirements are met. To operate at large scale and remain profitable while maintaining the flexible and affordable pricing model offered to users,

sustainable and efficient operations is required in cloud DCs. Therefore, improvements in cloud centric DCs in the last two decades are largely responsible for significant changes in DC environments.

2.3 Modern Data Centre Infrastructure

A common practise in most modern cloud DCs is the adoption of commodity server hardware instead of bespoke server hardware used in traditional DCs. This aided capital expenditure (CAPEX) and operational (OPEX) reduction because such technology is widely available and well understood. However, utilisation remained low because the capacity and agility of the DC was defined by the capacity of each physical server. Each physical server supported a single monolithic application. The advent of virtualisation at the beginning of this millennium enabled multiple applications to share a single physical server by running in independent virtual environments. Virtualisation decouples compute, network and storage resources from their underlying physical hardware. Virtualisation also enables on-demand allocation, de-allocation, and re-allocation of logical instances of these resources which are aggregated as virtual machines (VMs). A software or firmware called hypervisor which runs on physical resources is the entity that makes this abstraction possible. Wide adoption of virtualisation in DCs enabled improved utilisation (to about 50%) and energy efficiency. Thus, leading to reductions in CAPEX and OPEX of DC infrastructure providers and consequently reduced user's consumption bills.

Even though virtualisation enabled the pooling of compute, storage and networking resources to support on-demand provisioning for dynamic applications, DC network bottlenecks inhibited the maximal agility [21]. This is because of increased east-west traffic within virtualised DCs. Subsequent estimates by Cisco put such east-west traffic within DCs at 77% of global traffic in modern virtualised DC [4]. Hence, to mitigate link oversubscription resulting from increased east-west traffic in virtualised DCs, adoption of virtualisation was initially limited; consequently, resource fragmentation persisted in DCs.

The historical challenges of DC network fabrics were addressed via switch-centric and server-centric DC network topologies proposed by the industrial and academic communities. Switch-centric DC network topologies are hierarchical networks that adopt switches as relay nodes during traffic forwarding between DC components. Such network topologies scale by introducing additional switches at different layers. The fat-tree topology [22],

[23] is a cheap switch-centric topology proposed for the DC environment which adopts identical commodity switches in all three layers. The spine-and-leaf network topology has also been proposed to provide predictable low-latency communication in virtualised DCs with increased east-west traffic [24]. Spine-and-leaf topology is often a two-tiered network comprising of leaf switches in the lower tier and spine switches in the upper tier. In the southbound direction, the leaf switches connect to DC servers. While in the northbound direction, leaf switches connect to all spine switches to form a full mesh as shown in Figure 2.2. Occasionally spine-and-leaf topology may be implemented as a three-tier topology. Figure 2.2 shows a three-tier topology where an additional spine layer comprising of high capacity super-spine switches is present. The super-spine switches interconnect spine switches in large DCs and connect the DC to the Internet and other remote DCs. The spine-and-leaf network topology supports non-disruptive scaling via the addition of new switches and links. Server-centric DC network topologies which use servers as relay node for multi-hop communication have also been proposed. In addition to their computing functions, servers in such topologies are also equipped with multiple network interfaces. A portion of servers computing capacity is dedicated to packet processing and forwarding functions. However, the recent emergence of smart network interface cards (NICs) that can perform some or all network traffic processing in the server can minimise CPU usage in server centric topologies [25]. Examples of server centric topologies include Bcube [26], DCell [19] and FiConn [27].

Although, copper-based Ethernet links were deployed in traditional DCs, recent trends show that advances in silicon photonics technology has enabled the deployment of optical transceivers in DCs. Hence, optical links are being adopted to replace copper links in DCs [23]. Further use of optics in DC has also been proposed in other instances where optical components complement the electronic switching technologies in hybrid network topologies. Examples of such hybrid network topologies include C-through [28], Helios [29] and optical switching architecture [30]. All-optical network topologies such as Lightness [31] and datacentre optical switch [32] have also been proposed for DC environments. Motivating benefits of adopting optical communication in DC include, lower power consumption, longer transmission distances, lower latency, and higher capacity. For instance, passive optical network (PON) based optical topologies for DCs, which achieved significant power savings, were proposed in [33]. Figure 2.3 gives a summary of some network topologies proposed for modern DCs.

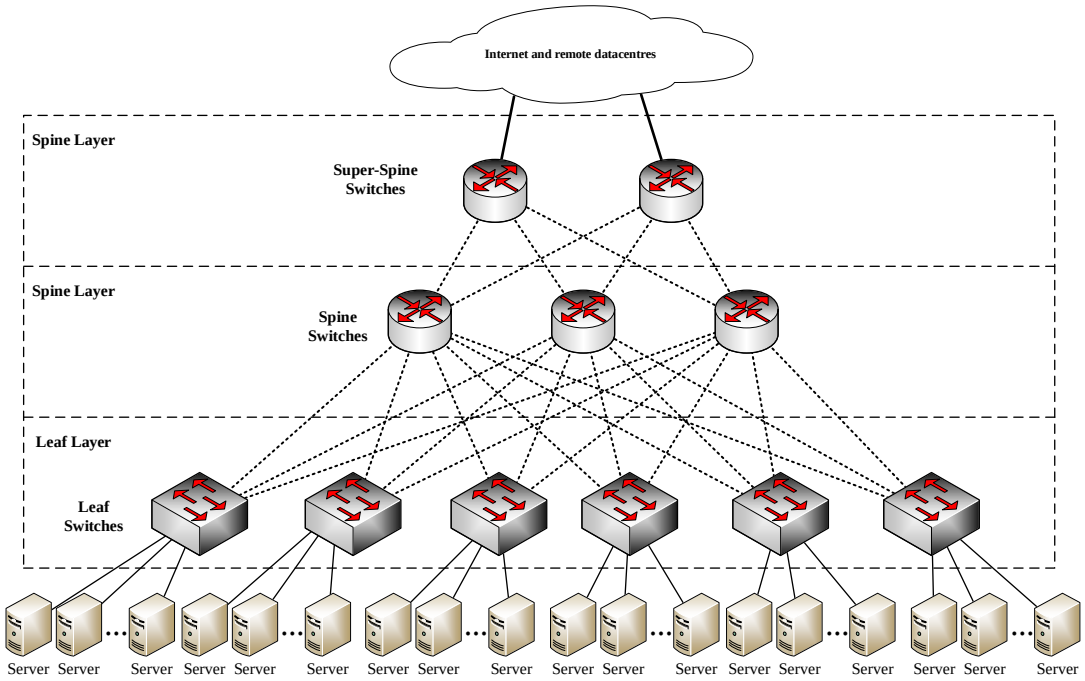


Figure 2.2: Three-tier spine-and-leaf data centre network topology

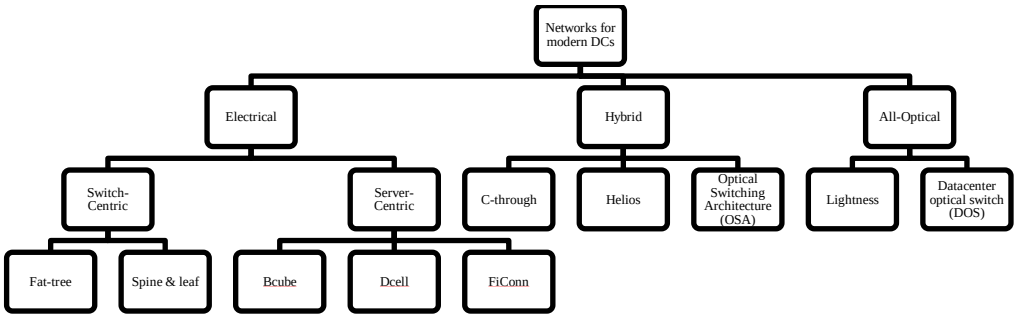


Figure 2.3: A summary of networks for modern datacentres

2.4 Techniques for Improved Efficiency in Data Centres

Modern DCs adopt a range of software centric and hardware centric techniques to improve efficiency as illustrated in Figure 2.4.

2.4.1 Software Centric Techniques

In addition to the adoption of virtualisation, today's modern DC adopt other software-based techniques. These techniques improve DC operational and energy efficiencies through concepts such as containerisation of applications, cloud native architecture for workloads and software defined infrastructure and serverless computing [34]. Prior to the wide adoption of containerisation in DCs, each VM supported a single application. Containerisation is an alternative or a complement of virtualisation. Containerisation enables multiple logical instances of homogenous or heterogeneous applications to run independently on the same VM or on a physical server. Each logical

instance of an application is called a container. A container is a packaged software that is abstracted from the operating system (OS) of a VM or server, comprising of the source code of an application and all dependencies required for the application to run on various physical and virtual platforms [35]. Unlike VMs, multiple containers can share a common OS in a given server. An open-source orchestration platform such as Kubernetes [36] can be used to manage multiple containers. Containerisation enables significant improvement in DC efficiency by reducing the number of servers required and by improving active server utilisation.

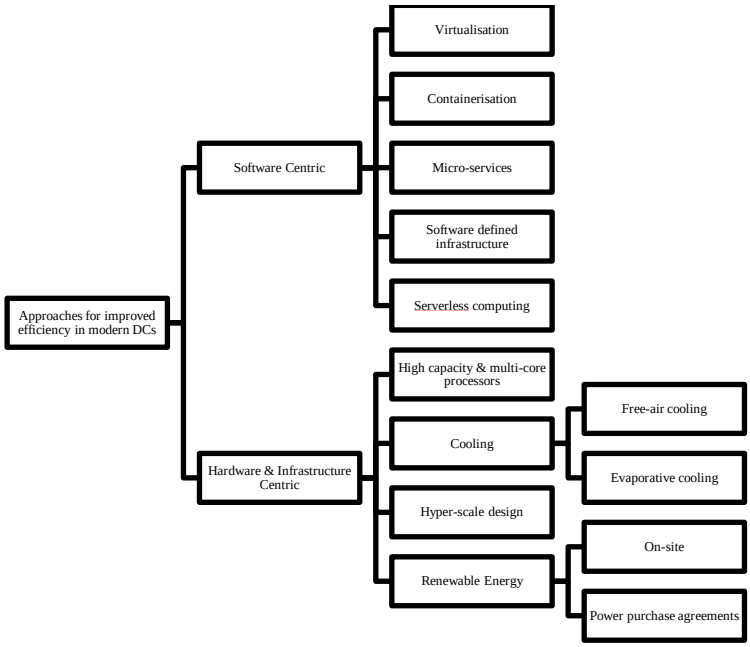


Figure 2.4: A overview of techniques for improved efficiency in data centres

The concept of containerisation has also promoted the improvements in application design architecture for greater efficiency and utilisation by supporting containerised micro-services of the cloud native application design. However, monolithic workloads can also be containerised. Traditionally, applications are designed as monolithic workloads. Monolithic workloads have fixed resource requirements and are designed to run on bespoke physical or virtual hardware built for the workload's peak performance. A monolithic workload comprise of multiple closely coupled components performing different business functions [37]. Thus, each component within the workload cannot scale independently and the development lifecycle is sequential. Therefore, monolithic workloads can only be scaled via replication of each workload onto multiple servers or VMs. This increases the quantity of hardware required. As an alternative workload architecture the cloud-native design approach proposes the decomposition of monolithic workloads into independent components called micro-services.

Each micro-service performs a specific business function that can be developed, tested, deployed, managed and scaled individually [38]. These related micro-services performing different business functions are thereafter combined to form an integrated application which is comparable to the decomposable monolithic workload. Communication between these loosely coupled micro-services, which are associated with the same integrated application, is facilitated using well-defined standards or application programming interfaces (APIs) [39]–[41]. The APIs isolate inner workings of each micro-service. This novel approach enables features such as massive scalability, agility, and improved resource utilisation (and the corresponding energy efficiency it enables) desired for distributed applications deployed in cloud computing DCs. Integrated applications scale by replicating only the micro-service implementing desired business functions; thus, enabling a new level of workload modularity.

DC operators have also adopted the concept of software defined infrastructure (SDI) to achieve holistic efficiency in the entire DC infrastructure. SDI uses automation enabled by software to control DC functions including deployment, provisioning, configuration, and operations. In addition to the virtualisation of servers, other components of the DC such as network, power and cooling infrastructure are also controlled via software. SDI enables centralised control of DCs to support on-demand rightsizing of the infrastructure to temporal workloads requirements. This enables new levels of agility and efficiency in the DCs which is expected to increase the utilisation of installed capacity to a maximum of 70% [42]. The SDI concept has also promoted the introduction of the serverless computing service by leading cloud providers [43]–[46]. The serverless computing paradigm provides computing capability as a managed service to users. Simultaneously, the cloud service providers utilise SDI capabilities at the backend to ensure that compute resources are provisioned optimally in real-time to fit exact resource demands of cloud user’s applications. This combination of SDI and serverless computing significantly increases the economic and environmental sustainability of DCs [47].

These software-centric approaches used in today’s DCs are sub-optimal relative to the expected future growth in computing capacity usage. Hence, the ratio of computing capacity to its corresponding cost remains low, primarily because of the architecture of traditional servers used in computing infrastructures. Therefore, other approaches are also being actively explored

by DC infrastructure providers to improve the economic and environmental sustainability of computing infrastructures.

2.4.2 Hardware and Infrastructure Centric Techniques

In the last decade, hardware and infrastructure centric measures have been taken to complement software centric approaches directed at improving power efficiency, cooling technologies and management of DCs. Forecasts, such as a prediction that DCs will account for 3-13% of global electricity consumption in 2030 relative to 1% in 2010 [48], are primary motivators for the development of these hardware and infrastructure centric approaches. Moreover, some characteristics of the traditional DCs have revealed the inefficiencies of DC infrastructure. For example, the traditional DC is known to have high power usage effectiveness (PUE). Hence, up to a third of the input power to the traditional DC is used for cooling alone [21]. Examples of hardware centric measures used to improve efficiency include the introduction of high-capacity multi-core processors with multi-threading features and lower power consumption. Hence, more computation is enabled at lower power consumption and low physical footprint. The thermal efficiency of such processors was also improved so that less cooling infrastructure is required in modern DC. Highly energy and water efficient evaporative cooling and free-air cooling method are increasingly being adopted to supplement or replace mechanical chillers in massive DC [49], [50].

Additionally, cloud giants such as Facebook have adopted the hyper-scale design in DCs to provide much needed agility, reduced total cost of ownership (TCO) and improved resource utilisation. Hyper-scale design adopts easily swappable commodity computing components in streamlined modular servers which are deployed in massive DCs. The open compute project (OCP) was formed to promote an open-source hardware ecosystem for hyper-scale DCs. OCP aims to re-design DC infrastructure using commodity hardware for increased energy and cost efficiency. An initial contribution to the OCP by Facebook enabled a DC that achieved an initial PUE ratio of 1.07 and was 38% more energy efficient while achieving a 24% reduction in operation cost [51], [52]. Intel, a contributor to OCP, has also proposed a reference architecture called the Rack Scale Design (RSD) [53]. Intel’s RSD simplifies the adoption of hyperscale design in small DCs deployed in enterprises and telecommunication industry. As a result, Intel’s RSD partners [54] such as Ericsson, Dell, HPE and Inspur have adopted the reference architecture to produce computing infrastructures that delivers benefits enjoyed by cloud giants to enterprise of various sizes. For example,

the network functions virtualisation (NFV) [55] can leverage on the benefits enabled by hyper-scale design for greater efficiency at the edge of the network. The NFV paradigm in the telecommunication industry promotes the virtualisation of network functions and components to run in commodity hardware.

Operators of massive DCs have also made significant efforts to reduce the carbon footprints of DC infrastructure in parallel with improvements in operational and energy efficiencies. To this end, the construction of cloud computing infrastructure near renewable energy sources (such as wind and solar) has been explored by DCs operators [56]. However, sole adoption of this approach can lead to performance degradation because temporal and geographical availability of renewable energy and user population distribution are not always in sync. Hence, cloud infrastructure providers are known to offset their carbon footprint via bulk purchase of renewable energy through power purchase agreements [57]. More recently, Google has committed to operate (all services including cloud computing) solely on carbon-free energy by 2030 [58].

In spite of significant improvements to traditional DC infrastructure, some shortfalls of the infrastructure persist. These include integration of resource and server proportionalities during infrastructure rollouts and upgrades; resource fragmentation and utilisation inefficiencies [59], [16]; high workload blockage probability [16], [60]; high infrastructure CAPEX and OPEX [61], [62]; and poor support for wide range of emerging applications (such as applications adopting in-memory computing architecture) that require ultra-low latency access to large data sets. The rigid architecture of today's DCs also limits the integration/adoption of advanced hardware technologies. For example, it is difficult to adopt novel storage class memory (SCM) [63] in today's DC. SCM have higher storage capacity and lower latency compared to traditional HDDs. As a result, disaggregation of computing servers has been proposed in recent times to complement other existing techniques that improve DC infrastructure efficiency.

2.5 Wavelength Division Multiplexing Networks

Wavelength Division Multiplexing (WDM) technology enables the maximisation of the usable bandwidth in an optical fibre by supporting concurrent transmission of multiple wavelengths over a single optical fibre. Each wavelength conveys independent signals at the maximum data rate supported by electronic processing speed. WDM technology is widely used in

communication networks including core, metro, and access networks. Traditionally, such networks deploy the WDM technology in a point-to-point system architecture where optical-electrical-optical (OEO) conversion is required at each intermediate node on an end-to-end communication path. However, advances in optical devices which support optical routing and switching in the optical domain, has enabled the adoption of WDM technology for networking [64]. In recent times, the WDM technology is also being deployed in the DCs where there is a strong desire to leverage on the benefits of optical technologies. This is expected to enable high capacity, low cost, and low power communication.

2.5.1 Optical Devices

A brief review of some optical devices that enabled the use of WDM technology for networking is as follows.

- **Optical fibre**: A high-speed medium that supports transmission in the optical domain. A fibre supports high transmission bandwidth. Optical fibres also support low attenuation transmission in certain areas of the spectrum where greater distances are covered by optical signal transmissions.
- **Coupler**: A coupler combines or splits signals in optical networks, it combines/splits signals from different sources into one or more output fibres. Couplers can be designed to be wavelength independent or wavelength selective.
- **Splitter and combiner**: A splitter is a passive device that splits optical signals carried by an optical fibre into multiple output fibres. For instance, a 1:N splitter splits optical signals conveyed by an optical fibre into multiple output fibres. On the other hand, a combiner combines optical signal from multiple optical fibres into one optical fibre.
- **Arrayed Waveguide Grating (AWG)**: A passive and data rate agnostic device used as a multiplexer or a de-multiplexer in WDM networks. As a multiplexer, the AWG combines a set of wavelengths into an optical fibre. On the other hand, as a de-multiplexer, the AWG separates optical signals on distinct wavelengths on an optical fibre into different output ports. A single AWG can function as a multiplexer in the forward direction and as a de-multiplexer in the reserve direction as a result of the cyclic property of the AWG.
- **Arrayed Waveguide Grating Router (AWGR)**: An NxN passive wavelength router capable of supporting contention free all-to-all communication between N nodes by using N wavelengths. To communicate with N nodes concurrently, each node requires N transceivers. AWGR has a fixed routing matrix determined by an inherent cyclic routing property. Hence, a wavelength that enters the device at a specific input port exits at a specific output port. A wavelength supported by the AWGR can be reused concurrently over

the routing fabric provided every active instance of the wavelength enters the device at a different input port. An AWGR can be adopted to implement an optical cross connect (OXC) in a WDM network, since it can facilitate optical signal switching without conversion.

- **Wavelength Selective Switch (WSS)**: An active optical component that can independently route any wavelength at its shared input port to any one of its multiple output ports. For instance, a 1xN WSS can switch a set of wavelengths at its common input port to any of the N output ports. Wavelength routing and switching of a WSS can be modified via an electronic control interface on the device. The WSS can also be adopted to implement an OXC in a WDM network.
- **Circulator**: A multiport device that allows optical signals to travel in one direction only. The device ensures that optical signals entering any of its ports leaves at the next port. Circulators can facilitate bi-directional transmission on a single strand of optical fibre.
- **Fibre Bragg Grating (FBG)**: A passive optical device that reflects an optical signal of a specific wavelength while allowing transparent transmission of other wavelengths. This is achieved by periodically changing the refractive index of the fibre core along the length of the fibre. The FBG can be made tuneable to control the reflected wavelength by stretching the FBG or by changing the temperature. FBG has a typical tuning latency in the milliseconds, for example 2 milliseconds as stated in [65].

2.5.2 WDM Network Architectures

There are two classes of WDM network architecture i.e., broadcast and select WDM networks and wavelength routed WDM networks.

2.5.2.1 Broadcast and Select WDM Networks

A common medium is shared in broadcast and select WDM networks and a simple broadcast mechanism facilitates the transmission and reception of optical signals between network nodes. The star topology is a popular example of the broadcast and select WDM network. In the star topology, a passive coupler is the hub to which all network nodes are connected. The coupler combines the wavelength transmitted from each network node in the topology and forwards it to all other network nodes. Each node tunes its receiver(s) to the wavelengths intended for it.

2.5.2.2 Wavelength-Routed WDM Networks

Circuit-switched light-paths are established between communicating pairs of nodes in a wavelength-routed WDM network architecture. A light-path is a directed all-optical connection between a pair of nodes. It may traverse multiple optical fibres and may use multiple wavelengths without undergoing OEO conversion at intermediate nodes. In wavelength-routed WDM networks

establishment of light-paths is controlled by certain constraints such as wavelength distinct constraints, wavelength reuse property and wavelength continuity.

Wavelength distinct constraints ensure that each active wavelength in an optical fibre link is unique. The wavelength reuse property of wavelength routed WDM network architecture enables the reuse of a given wavelength on disjoint optical fibres. Hence, space division multiplexing (SDM) can complement WDM to increase network throughput. In the absence of wavelength conversion in the optical domain, a light-path must use the same wavelength on all links traversed; this is the wavelength continuity constraint in WDM networks which may limit wavelength utilisation and connection request satisfaction in the network.

2.5.3 Virtual Topology Design Problem in WDM Networks

Given the physical topology of a WDM network, network limitations and the connection request between pairs of nodes, the task of creating a set of light-paths over the physical topology to support traffic demands while optimising network performance is called a virtual topology design problem [64]. Virtual topology design problem can be formulated as a MILP problem. Examples of network limitations includes number of wavelengths available on an optical fibre link, the number of transceivers available at each node. Network performance metrics may include end-to-end delay, congestion, and network power consumption.

A connection request between a pair of nodes in a WDM network may be provisioned by single light-path or by multiple distinct light-paths on an end-to-end communication path. When multiple light-paths are provisioned to satisfy a network connection request, OEO conversion is performed at the terminal node of each light-path (at intermediate nodes between two light-paths). OEO conversions at such intermediate nodes can increase network performance in terms of throughput, utilisation, and availability. However, this is achieved at the cost of additional processing and queuing delay on the end-to-end communication path.

The virtual topology design problem entails the creation of light-paths, subsequent routing of the light-paths over the physical topology, the assignment of wavelengths to each light-path and finally, the routing of connection requests over all created light-paths. The virtual topology forms a transparent optical layer that lies between the physical layer and higher layers (i.e., IP, SDH/SONET and ATM) of the WDM network topology. Some network

design problems considered in this thesis are virtual topology design problems and are formulated as MILP models.

2.6 Mixed Integer Linear Programming

A linear programming problem aims to optimise (either maximise or minimise) a linear objective function subject to linear constraints. The optimal solution of a linear programming problem lies in the feasible region bounded by the linear constraints of the problem [66]. The general form of a linear programming problem comprises of parameters, variables, constraints, and an objective function [67].

- **Objective function**: This is the optimisation goal of a linear programming problem; it is a linear equation which comprises of variables that are optimised to obtain the optimal solution.
- **Variables**: These are unknown values or decision variables that are optimised to obtain an optimal value that satisfies the linear constraints of the linear programme.
- **Constraints**: These are linear equalities and/or inequalities that limit the values taken by decision variables. The constraints of linear programme problem collectively define the feasible region where an optimal solution can be found for the problem.
- **Parameters**: These are known values that are given as an input to the linear programme. They represent costs and (lower or upper) thresholds which are associated with variables, constraints, and the objective function of a linear programme.

Along with linear objective function and constraints, all variables of a linear programme take real values by default. However, when some variables take integer values, the problem is an integer linear programme (ILP). Another variant of linear programmes where some variable take integer values while other variables take real values is a MILP. For example, given, the demand volume and the cost of forwarding traffic via each link in the network, an un-capacitated network design problem that minimises total network cost can be represented by the following MILP model as given in Chapter 2 of [67].

Set and Parameters

D	Set of demands
E	Set of links
P	Set of paths in the network
ξ_e	Cost incurred on the link $e \in E$

h_d Demand volume $d \in D$

Variables

x_{dp} The flow of demand $d \in D$ on path $p \in P$

δ_{edp} $\delta_{edp} = 1$ if a flow of demand $d \in D$ on path $p \in P$ traverses link $e \in E$. Otherwise, $\delta_{edp} = 0$

y_e Unknown capacity of link $e \in E$

Objective: Minimise:

$$\sum_{e \in E} \xi_e \, y_e \tag{2.1}$$

Equation (2.1) is the objective function, it minimises the total network cost.

Subject to:

$$\sum_{p \in P} x_{dp} = h_d \tag{2.2}$$

$$\forall \, d \in D$$

Equation (2.2) are the demand constraints.

$$\sum_{d \in D} \sum_{p \in P} \delta_{edp} \, x_{dp} \leq y_e \tag{2.3}$$

$$\forall \, e \in E$$

Equation (2.3) are the capacity constraints.

Network design and resource assignment problems in DCs can be formulated as MILP problems which yield exact results. However, MILP problems are known to be NP-hard and are consequentially computationally intractable [64]; therefore, it is unsuitable for application in online scenarios and in large networks and DC scenarios. Heuristic algorithms that mimic the MILP solution are often designed to obtain approximate optimal solution for such a problem. Additionally, an analysis of an (approximate) optimal solution from solving MILP programme for small scenarios can provide insights to aid the design of heuristic algorithms. These algorithms can then be adopted for larger on-line scenarios. This approach is adopted in this thesis, where MILP problems are formulated for the workload resource demand placement in DCs and network connection request satisfaction. The formulated problems are evaluated under small illustrative scenarios and the exact results obtained are analysed to derive useful insights/patterns. These insights guide the design of heuristic algorithms that mimic the MILP approach to quickly obtain approximate optimal solutions for large practical scenarios. The MILP

equations in this thesis are coded using AMPL (A Mathematical Programming Language) [68] and solved with the IBM ILOG CPLEX solver [69] on the ARC3 supercomputing nodes with 24 CPU cores and 128 GB of memory [70].

2.7 Non-Linear Programming and Metaheuristics

2.7.1 Non-Linear Programming

Unlike linear programming problems where both objective function and constraint functions are linear, many practical problems are non-linear. A problem that has non-linear objective function and/or non-linear constraint function(s) is called a non-linear programming problem. There are different classes of non-linear programmes. The class of a non-linear programming problem depends on the features of its objective function and the constraints [66]. Classes of non-linear programming problems are as follows:

- **Unconstrained optimisation problems**: Problems in this class of non-linear programming problems do not have constraints, they only have a non-linear objective function.
- **Linearly constrained optimisation problems**: Linearly constrained optimisation problems have a non-linear objective function, but all the constraints are linear.
- **Convex optimisation problem**: The constraints of a convex optimisation problem are all convex functions while the objective function is a concave function if minimising or is a convex function if maximising. The feasible region of a convex optimisation problem is a convex region.
- **Non-convex optimisation problem**: A problem that has a non-convex objective function and/or one or more non-convex constraints. Non-convex programming problems are unable to guarantee that a local maximum will also be a global maximum.

It is important to note that a concave function always curves downward while a convex function always curves upwards. A function that curves upwards and downwards is a non-convex function. There are few instances of non-linearity in this thesis. However, such instances of non-linear constraints are adequately reformulated as linear constraints. Hence, all problems formulated in this thesis are linear programming optimisation problems.

2.7.2 Metaheuristic Optimisation Methods

In contrast to the design of targeted heuristics for the specific problems formulated in the thesis, more generic metaheuristics can also be considered. Generic metaheuristics are effective at finding global optimal solutions for non-linear programming problems. Simulated annealing and particle swarm optimisation are good examples of such metaheuristic optimisation methods.

2.7.2.1 Simulated Annealing

Simulated annealing is stochastic optimization method that is based on the analogy to the physical annealing process. Given enough time, the approach adopted in simulated annealing can yield a global optimum solution because local optimum solutions are escaped [66]. This is achieved by always accepting a better solution while also occasionally accepting worse solutions with a certain acceptance probability. In early stages of the running the algorithm, this acceptance probability is high but decreased as iterations of the algorithm increases. Hence, more solutions are explored in the early stages but the algorithm becomes conservative over time by tending to accept only better solutions. Generally, the functions the determines the acceptance probability for worse solutions encourages acceptance of worse solutions that are relatively close to the best know current solution. Acceptance of solutions that are significantly worse-off are discouraged by the function.

2.7.2.2 Particle Swarm Optimisation

Particle swarm optimisation (PSO) algorithm is a population-based stochastic optimisation method modelled after swarming and flocking behaviour in animals. PSO algorithm has a swarm (i.e., a constant population) of particles. Each particle is a possible solution to the optimisation problem. A particle moves through the search space by changing its position based on its own experience and the experience of its neighbours [71]. In each iteration of the PSO algorithm, a particle is accelerated towards its best-known position, since the algorithm began, and towards the best-known position in its neighbourhood [72]. A best neighbourhood position is determined by social information exchange in that neighbourhood. PSO is a simple and computationally efficient algorithm that effectively escapes local optimum solutions in the search for the global optimum.

This thesis does not adopt any metaheuristic optimisation method. As an extension of the work in this thesis, metaheuristic optimisation methods may be used to validate the MILP and targeted heuristics proposed later in the thesis. Additionally, other metaheuristic optimisation methods such genetic

algorithm [66], [71] and ant colony optimisation [73] can also be adopted to solve both linear and non-linear programming problems.

2.8 Summary

This chapter presented an overview of the evolution of DC infrastructure and DC networks while highlighting the historical challenges of DCs and their intrinsic networks. We have also given an up-to-date review of the software-centric and hardware/infrastructure-centric approaches that have been employed to improve the efficiency in DC infrastructure. Furthermore, we have presented a brief overview of WDM networks and enabling optical components and architectures. Furthermore, we described the virtual topology design problem in WDM networks which can be formulated as a MILP model. We gave an introduction of the MILP optimisation approach that is adopted to formulate workload demand placement in DCs and network connection request problems in this thesis. Finally, we also gave a brief review of non-linear programming problems and metaheuristic optimisation methods. The discussions in this chapter give background knowledge and serve as a motivation for the concepts considered and methods adopted in the remaining Chapters of this thesis.

Chapter 3 : Review of Composable Data Centre Infrastructure

3.1 Introduction

The work presented in this thesis evaluates the energy efficiency of composable DC infrastructure and the application of the disaggregation concept in the fog computing tier. This chapter presents an overview of composable DC infrastructure. We also briefly review the three primary enabling technologies for composable DCs i.e., resource disaggregation, software defined infrastructure and optical communication and silicon photonics. Furthermore, we highlight the primary benefits that composable DCs will enable and also discuss some implementation challenges associated with composable DCs. Finally, we present a review of composable DCs research and projects in the literature.

3.2 Composable Data Centre Infrastructure Overview

Composable data centre (DC) infrastructure deploys software driven techniques for on-demand composition, decomposition, and re-composition of logical computing nodes to support applications. This is achieved via the orchestration of physical or logical pools of disaggregated resource components that are inter-connected over appropriate networks.

3.3 Enabling Technologies

The key enabling technologies of composable DC infrastructure are resource disaggregation, SDI, optical communication, and silicon photonics.

3.3.1 Resource Disaggregation

Resource disaggregation is an essential enabling concept in composable DC infrastructure. It addresses the resource stranding problem in today's DCs. Resource stranding is the primary cause of the poor utilisation efficiency associated with traditional servers in today's DC. The problem occurs because the chassis of a traditional server acts as physical utilisation boundary for the server's intrinsic resources. Hence, if the capacity of a unique resource type within a given server is unable to support additional application consolidation, components of all other unique resource types also become unavailable irrespective of each resource's idle capacity. Rack 1 in Figure 3.1 shows the impact of resource stranding as input applications A-G are provisioned. The

introduction of resource disaggregation mitigates this problem in computing infrastructures.

Disaggregation of server hardware resource components can be performed under different utilisation scopes (as illustrated in Figure 3.2) to achieve the intended benefits. The utilisation scope of a disaggregated resource component refers to the maximum utilisation boundary which must be maintained when the component is selected to form a logical server. This scope can be controlled physically or logically. Additionally, both approaches can be combined to achieve hybrid disaggregation.

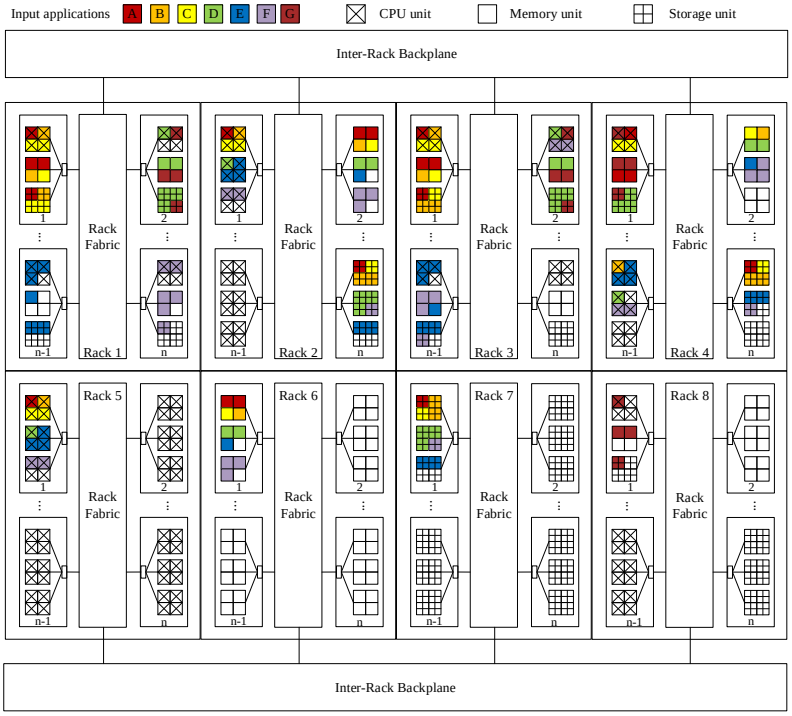


Figure 3.1: Resource disaggregation in composable DC infrastructure

This figure shows various forms of resource disaggregation in a composable DC infrastructure. Rack 1 comprise of traditional servers that do not implement resource disaggregation. Resource components in Rack 2 are physically disaggregated at rack-scale; the rack comprises of homogeneous compute nodes. Resource components in rack 3 are logically disaggregated at rack-scale. Racks 4 and 8 implements hybrid disaggregation; 1 compute node in Rack 4 is heterogeneous while others are homogenous. Rack 5, 6, and 7 jointly implement physical disaggregation at pod-scale.

Table 3.1: Infrastructure template of applications

This table gives the infrastructure template of input applications A-G. The infrastructure template of an application is the logical host configuration needed to support that application. Each infrastructure template may have CPU, memory and storage resource demands and an abstracted service level agreement (SLA) for the optimal performance of the application it supports. The maximum tolerated intra-logical server latency between the resource components of a given application represents the SLA requirement of each application.

Apps	CPU units	RAM units	Storage units	Intra-logical server latency
A	1	2	1	Pod
B	1	1	2	Pod
C	2	1	1	Pod
D	1	2	3	Pod
E	3	1	2	Pod
F	2	3	1	Pod
G	1	2	1	Node

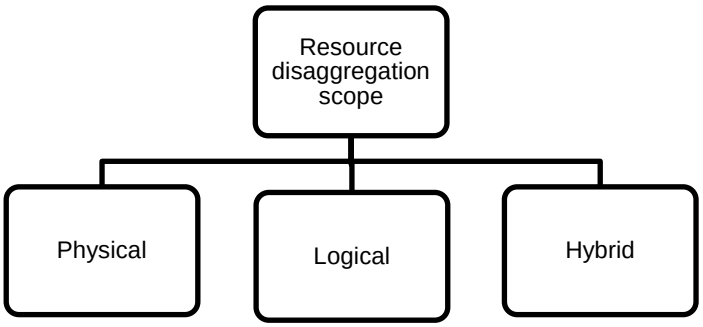


Figure 3.2: Scopes of resource disaggregation

3.3.1.1 Physical Disaggregation

Physical disaggregation entails physical separation of computing resource components into pools of homogenous resource types. A homogenous resourced pool can take the form of a server-like node or a sled which can fit into a standard rack-chassis. Such a node comprises of one or more components of the same resource type and it is the basic unit of all physically disaggregated composable DCs as shown in Figure 3.1. The utilisation scope of each homogeneous resourced pool can be node-limited, rack-limited, or pod-limited relative to the size of the DC under consideration. The allocation of heterogeneous nodes and their corresponding utilisation scope determine the scale of physical disaggregation in composable DCs as shown in Figure 3.1.

The scales of physical disaggregation are rack-scale, pod-scale and DC-scale [62], [17]. At rack-scale, the composable DC comprise of multiple racks, each rack comprises of multiple (homogeneous) nodes of different resource types and the utilisation scope of each node in each rack is rack-limited. Rack 2 in Figure 3.1 represents a physically disaggregated composable DC at rack-scale which is used to provision input applications A-F. At pod-scale, the composable DC comprises of multiple racks, each rack comprises of multiple homogeneously resourced nodes and the utilisation scope of each node in each rack is pod limited. Racks 5-7 of Figure 3.1 shows physical disaggregation at pod-scale where applications A-F have been provisioned. Finally, at DC-scale, the composable DC comprises of multiple racks, each pod in the DC comprises of homogenously resourced racks, and each rack comprises of multiple homogeneously resourced nodes.

3.3.1.2 Logical Disaggregation

In contrast with physical disaggregation, the utilisation scope of logically disaggregated resources is not controlled by physical boundaries within the DC. Rather, boundaries are imposed virtually and on-demand by a central control entity with global knowledge of the infrastructure's state. Logical disaggregation allows the reuse of traditional server chassis architecture as shown in Rack 3 of Figure 3.1. Hence, a node in a logically disaggregated computing infrastructure may comprise of heterogeneous resource components like traditional servers. However, it is important to that hyper-scale design principles be adopted for each physical pool of homogeneous resources in a given node within the rack. This will ensure easy swapping/upgrade/replacement of modular resource components if required.

3.3.1.3 Hybrid Disaggregation

A composable DC infrastructure that adopts the hybrid disaggregation approach combines both physical and logical disaggregation concepts. This helps to achieve greater efficiency without violating applications quality of service requirements. Hence, logical servers can be created to support application specific requirements. For example, multiple applications to be hosted in the same composable DC may have different intra-logical server latency requirements as illustrated in Table 3.1. The setup in Rack 4 of Figure 3.1, comprising of a mix of homogenously resourced nodes and heterogeneously resourced nodes implements hybrid disaggregation. This setup enables a composable DC capable of supporting all input applications listed in Table 3.1 with optimal efficiency. This contrasts with the situation when physical disaggregation is performed at rack-scale or pod-scale where

input application G must be rejected. This rejection is because of the violation of the intra-logical server latency of App G when physical disaggregation is employed at rack-scale or pod-scale. This problem is easily resolve by introducing Rack 8 as shown in Figure 3.1.

3.3.2 Software Defined Infrastructure

A software defined infrastructure (SDI) is an infrastructure that can mutate on-demand to simultaneously satisfy changing users' demands and its provider's desires by leveraging deep monitoring and underlying infrastructure capabilities [74]. Softwarisation and automation of underlying infrastructure's hardware resources, such as compute, storage and network lie at the heart of SDI. These resources are softwarised independently and become programmable. Consequently, software defined compute (SDC), software defined storage (SDS) and software defined network (SDN) are achieved within the DC infrastructure. A complete SDI is further supported by centralised controllers, deep monitoring of underlying resources and a management sub-system. In SDI, the creation, scaling and/or deletion of softwarised resources and infrastructure are automated as required to achieve desired objectives.

However, a precursor of dynamic transformation of softwarised resources in SDI is the abstraction of such resources from their physical hardware through virtualisation. Virtualisation enables the creation of logical resources from physical resources. These logical resources are subsequently aggregated to form logical servers that can support applications seamlessly. In the computing domain, virtual compute resources may be instantiated as a bare metal, a VM or a container. SDC dynamically controls the instantiation, termination, and migration of virtual computing infrastructures to satisfy user demands and provider’s constraints. SDS provides similar functions for virtual storage in the SDI. These virtual storage resources are created from abstracted heterogeneous storage components which are controlled centrally. The logical resources of a virtual infrastructure are connected by virtual networks. Theses virtual networks are overlaid on underlying physical networks to satisfy the performance service level agreements (SLAs) of virtual infrastructures. Furthermore, virtual networks in SDIs also provide communication paths between logical infrastructures and between the logical infrastructures and other external systems. SDN implements dynamic mutation of virtual networks to ensure continual satisfaction of desired objectives in the SDI.

Resource softwarisation entails the separation of the control and data planes of the corresponding resource. An independent control plane which is unique to each resource type (i.e., SDC controller, SDS controller or SDN controller) is responsible for the coordination and orchestration of the underlying data plane of that unique resource type. This the coordination and orchestration creates virtual instances of the resource via standard interfaces. On the other hand, the data plane of a given resource type comprises of one or more components of the resource. A central controller of the SDI system coordinates all independent resource controllers in the infrastructure to ensure that SLAs of virtual infrastructures are always satisfied. The SDI controller also performs analytics on data collected by the monitoring subsystem to derive proactive and reactive triggers for system mutation. Policies to set-up the SDI in the most efficient way using insights from historical knowledge and best practices can also be implement at the SDI controller in conjunction with the underlying resource controller.

SDI is essential to all software-oriented approaches focused on increasing cost and energy efficiencies in today's computing infrastructures. Before the advent of composable infrastructure, SDI enabled the implementation of these software-oriented approaches by providing functions such as on-demand scaling of virtual resources; deep monitoring of physical and virtual resources relative to defined SLAs; intelligent transformation of underlying infrastructure based on insights from monitoring data analytics; on-demand mapping of virtual resources to physical resources; and the use of knowledge base to improve performance and operations [74]. Composable DC infrastructures leverage and extend these functions as required to support the delivery of greater efficiency.

3.3.3 Optical Communication and Silicon Photonics

Disaggregation of computing resources in composable DC brings about the need for a high bandwidth, low latency, flexible and energy efficient interconnection between disaggregated resource-components. The many benefits of optical networking technologies (such as low latency, low interference, long-distance, high bandwidth, low energy, and scalable communication) make them better candidates over electronic networking technologies for such a network. However, buffering in the optical switching domain remains challenging, especially in electronic-based computing infrastructures where network congestion occurs frequently. Furthermore, historically, optical network components often have a large footprint and their manufacturing process is expensive. This is in contrast to the cheap and

matured manufacturing process of silicon micro-electronics circuits used in electronic network technologies.

The advent of silicon photonics technologies enabled the emergence of affordable hybrid networks designed using novel opto-electronic circuits. These circuits, which integrate electronics and photonics into a single circuit, are manufactured using cost effective and matured traditional silicon IC manufacturing processes [75]. Hence, opto-electronic networks that combine features of both electronic and optical network technologies are enabled. The resulting hybrid network addresses some challenges of electronic network technologies and avoids challenges of pure optical network technologies while leveraging the advantages of both. This is achieved through integrated silicon photonics components manufactured as photonic integrated circuits (PICs). PICs can effectively meet capacity, flexibility, energy efficiency, and scalability required in networks of composable DCs.

The use of silicon photonics technologies is not new, as it is widely used in optical transceivers in today's DCs and data communication networks. Attractive features of silicon photonics such as dense integration, higher data rates, long reach, low energy per bit (few pJ/bit) and low manufacturing cost makes them the preferred choice for next generation information and communication systems such as composable DCs. Forecasted drop in the cost of silicon photonics-based interconnects to $1/Gbps is also expected to further encourage the adoption of the technology [76]. Silicon photonics enabled opto-electronic networks are widely seen as the choice technology to deliver the connectivity required between disaggregated resource components [62], [77].

Furthermore, the emergence of densely integrated silicon photonic switches is expected to support the commercial implementation of composable DCs. For example, densely integrated hybrid packet switching devices, which co-package optics and ASIC switch chips, can be used in composable DCs to implement switches with optical IO. Such devices can also be used to design high-speed coherent integrated circuits or to manufacture system on chip (SoC) with integrated fabric switch [78]. Application of silicon photonics in on-board communication can also aid composable DCs. On-board communication can facilitate high speed and power efficient optical IO between switch chips and NICs and between resource components in nodes of a composable infrastructure. However, there are several challenges that inhibit effective commercialisation of silicon photonics technologies which will also affect their adoption in composable

DCs. These includes low power optoelectronics conversion integrated with passive MUXs and DEMUXs, low cost packaging, low loss and energy efficient on-chip lasers with low heat dissipation, power efficient modulators, integration of CMOS and photonic devices and low insertion loss and polarisation independence fibre attach [75], [76].

3.4 Benefits of Composable DC Infrastructure

The composable DC infrastructure offers several benefits which can enhance the efficiency of computing infrastructure. Such benefits are not limited to cloud DCs alone as they can also enhance deployments of commodity hardware to support network function virtualisation (NFV) implementation of telecom operators at the edge of the network and on premise hyper converged infrastructure (HCI) or DC of enterprises. The benefits of composable DC infrastructure include increased modularity, greater agility and flexibility and improved efficiency of computing infrastructure.

3.4.1 Modularity

The shift from server-wise resource utilisation to component-wise resource utilisation in composable DCs enables increased modularity and promotes proportional usage of underlying resource components. Furthermore, the modular design of underlying resource components in composable DC also enables precise upgrade and replacement of resource components. This contributes to the improved efficiency enabled by composable DCs.

3.4.2 Agility and Flexibility

Intelligent control and deep monitoring of the underlying (hardware) resource components of composable DCs supports greater agility and flexibility. The ability to guide automated transformation of the infrastructure on-demand with insights derived from intelligent analysis of collected monitoring data is responsible for this. On-demand slicing and aggregation of physical and virtual resource-components in composable DC enables an adaptable infrastructure that can scale dynamically at run-time based on temporal resource-demands of applications. Applications can also be migrated on-demand to ensure optimal performance via the automation of the composable DC. The adoption of multiple tiers of abstraction in composable DC effectively makes the infrastructure application agnostic despite the heterogeneity and complexity of the underlying hardware layer.

3.4.3 Greater Efficiencies

Composable DCs enable greater efficiency in multiple domains; hence, this is the most appealing benefit of the concept. Component-wise resource utilisation in composable DCs enables proportional usage of resource capacity which leads to higher resource utilisation efficiency relative to the modern DC. Therefore, the number of active resource components and total power consumed in composable DC are expected to be lower. This leads to improved energy efficiency because only necessary components are turned on while other components remain in an inactive state. Thus, the operational expenditure (OPEX) of infrastructure providers is curtailed. The number of required resource component in a composable DC is relatively lower than that of the modern DC due to increased utilisation of active resource components. Hence, capital expenditure (CAPEX) is reduced because fewer components are purchased, and the footprint of primary and auxiliary infrastructures is also reduced. Increase in the precision of upgrading and replacing modular resource components of composable DC can lead to further CAPEX reductions.

The adoption of software to control programmable underlying hardware components on-demand enables service-oriented consumption of computing capacity. This supports dynamic repurposing of idle compute capacities to provision other applications. It also promotes greater efficiency and generates revenues that can help infrastructure providers to offset their TCO. Adoption of in situ data processing in composable DC also enables greater network efficiency in instances where in-memory communication or data sharing is required. This is because data is processed at its location; hence, transmission over communication networks is minimised.

3.5 Implementation Challenges

Relative to state-of-the-art DC infrastructures, the concept of composable DC infrastructure introduces a range of challenges that must be addressed. Such challenges include but are not limited to the domains of physical networks, SDI, high availability, and application development and design.

3.5.1 Physical Networks

Exposure of essential intra-logical host communications onto higher tiers of DC networks introduces several challenges that inhibit full implementation of the composable DC. As illustrated in Table 3.2, these challenges are primarily associated with the CPU-memory and CPU-CPU communications. These

types of communication require ultra-high communication bandwidth and ultra-low latency for efficient performance of some applications. The physical limitations of traditional network media (such as fibre optics and copper media) control the maximum practical distance between disaggregated resource-components which form a logical host. Therefore, they control the scope and scale of resource disaggregation in composable infrastructure. However, propagation delay has minimal impact when optical fibre media are adopted. This is because of shorter communication distances in DC environments relative to the high speed and low attenuation of light propagation in optical fibre. Although, the adoption of optical communication provides a solution to many network challenges of composable DCs, today's optical technologies are somewhat limited [17]. For instance, if optical DC interconnects must stay competitive against electronic DC interconnects, the targeted energy efficiency values in Table 3.3 should be achieved at the different hierarchies of DC interconnect [79], [80]. Furthermore, the traditional network protocols and software stack [78] also introduces additional delays on the communication path between source and destination resource-components.

Table 3.2: Bandwidth and latency requirements for inter-resource communication in DCs[14]

Communication Type	**Bandwidth**	**Latency**
CPU – CPU	200 – 320 Gbps/CPU	10 ns
CPU – Memory	300 – 800Gbps/CPU	10 – 50 ns
CPU – Disk	5 – 128 Gbps/device	1 – 10 us

Table 3.3: Targeted energy efficient in future DC interconnects[79], [80]

DCN hierarchy	**Range**	**Energy per bit**
Core to core	< 1 cm	< 0.01 pJ/b
Chip to chip	1 - 5 cm	< 0.1 pJ/b
Module to module	5 – 30 cm	< 0.5 pJ/b
Board to board	0.3 – 1 m	< 1 pJ/b
Rack to Rack	1 -2000 m	< 1 pJ/b
Inter DC	1 - 100 km	< 10 pJ/b[80]

Parallel high-speed optical communication paths can be formed to ensure enough capacity for communication between physically disaggregated resource-components that form a logical infrastructure. However, integration

of multiple ports and switches onto resource-components in such a scenario introduces additional challenges. Adoption of silicon photonic technologies promises potential solutions to these challenges. However, as the number of disaggregated resource components increases in composable DC, the design of scalable physical network topologies must consider some criteria to find practical solutions. Such criteria could include flexibility, cost, power consumption and port/interface count.

The traditional network software stack can be streamlined to minimise additional delays introduced by the stack. However, this may be sub-optimal to satisfy the ultra-low latency communication between CPU and memory/CPU as required by some applications. A complementary solution could use the network software stack to relax inter-resource latency requirement by adopting techniques that amortise the latency related performance penalties in composable DCs. Furthermore, revisions in component hardware architecture may also be adopted to mitigate network challenges introduced by disaggregation. For instance, the use of intermediate hardware buffers or memory/data stores could be standardised in the physical architecture of composable DCs as observed in some prototypes of partially disaggregated DCs [61], [62], [81]. In this prototypes, primary and secondary memory tiers are implemented to minimise performance degradation. Storage-class memory (SCM), which offer higher capacity relative to DRAM and faster access speeds relative to HDDs, are suitable candidates to implement the secondary memory tier. DRAMs can remain in the primary memory tier [63].

Another network challenge which emerges with the advent of composable DC is the heterogeneity of interconnect interfaces between disaggregated resources. The heterogeneity of interfaces is expected to increase as the number, type and versions of resource-components increases in composable DCs. The proposition of a universal interface for resource-components in composable DC by the Gen-Z consortium [82] is a step in the right direction to address this problem. Going forward, such universal interface can be standardised for components deployed in composable DCs. Amidst several physical challenges of the networks of composable DC, exploring techniques to achieve greater flexibility and scalability over physical network topology of the infrastructure is also important. This is crucial in situations where slicing, aggregation and sharing of physical resource components is required. Hence, techniques such as network virtualisation and SDN should

be deployed to ensure maximum scalability of the physical network topologies in composable DCs.

3.5.2 Software Defined Infrastructure

Software is required to manage and orchestrate disaggregated resource-components which are transformed on-demand to create logical infrastructure. Beyond the regular features of conventional SDI, such software must support on-demand slicing and aggregation of underlying resource components to satisfy user demands. Therefore, relative to the conventional SDI, the demand for increased granularity in the control and management of composable DC introduces increased complexity and additional overheads to its management, operation, and maintenance.

A technique to simplify the complexity is the use of additional tiers and abstraction in the management and control system of composable DCs. Such an approach may lead to further loss in low-level control of underlying infrastructure which have been significantly abstracted. Furthermore, the SDI controllers in conjunction with lower-level controllers must implement policies that ensure optimal functioning of the composable DC. An example of such policy could be an algorithm which ensures optimal placement of applications demands in composable DC in real-time. Solving similar problems in traditional DC is difficult. This difficulty increases further in composable DC where component-wise resource consumption replaces server-wise resource consumption.

3.5.3 High Availability

The difficulty of achieving high availability (HA) increases in composable DC relative to modern DC. This is because HA must be implemented in both logical and physical layers of the data-plane (hardware) to ensure that the system is reliable and fault tolerant. For example, the process of creating a back-up (logical) server must ensure that selected components are physically disjoint from components that form the primary (logical) server to create a fail-safe system. Hence, the management and orchestration software of composable DC must support redundancy across physical and logical hardware domains. This introduces additional complexity to the management and orchestration software of the composable DC.

3.5.3 Application Design and Development

The emergence of composable DCs motivates re-evaluation of application design and development process to study the impact of the concept on the performance and suitability for legacy, present and future applications. Legacy

applications may fail to run on composable DC if the infrastructure design requires the optimisation of applications for the novel computing infrastructure platform. On the other hand, optimal efficiency of composable DC may be inhibited if the details of disaggregation are over-abstracted from the applications running on the platform [34]. Hence, the right balance must be struck between applications design and development process and infrastructure details abstraction.

3.6 Review of Composable Data Centres Research

In the past few years, the promise of improved modularity, agility, flexibility, and efficiencies has motivated numerous industry and academia led research into composable infrastructure. While some research projects are holistic and aim to achieve go-to-market products, other projects aim to address challenges or demonstrate the efficacy of composable infrastructures.

The concept of composable DC infrastructure has been explored by notable vendors in the IT infrastructure industry such as HPE [83] and Cisco [84]. Their primary aim was to create a programmable infrastructure that can simultaneously support legacy enterprise applications used to ensure efficient operations of businesses and emerging applications. The integration of CPU and memory resources into a common compute module i.e., partial disaggregation implies that efficient utilisation of CPU resources is constrained by the utilisation of the main memory resources or vice versa. This becomes pronounced when applications are memory or CPU intensive. Complete separation of CPU and memory resource i.e., full disaggregation requires an ultra-low latency and high capacity fabric which are difficult to implement using today's optical communication technologies [17]. To mitigate this problem, other composable infrastructure solutions [14], [81], [85]–[89] often settle for partial disaggregation. In partial-disaggregation, shared-independent high capacity memory modules are introduced to provide remote memory in the DC. Concurrently, the compute module's dedicated main memory component is maintained to function as a large cache. A comparison of DCs that adopt traditional server, partial disaggregation and full disaggregation at rack-scale was performed using a first fit algorithm in [17]. Scenarios with and without optical interface capacity constraints were considered for all DCs. The results showed that limited optical interface capacity inhibits optimal performances in full disaggregated DCs. Hence, traditional, and partially disaggregated DCs outperformed full disaggregation DCs. However, in the absence of such network bottlenecks, full

disaggregation performs better than partial disaggregated DC and traditional DC architecture.

Intel has also proposed a reference model called Rack Scale Design (RSD) [81], [85] for rack-scale composable infrastructure by adopting partial resource disaggregation. The reference model intends to reduce CAPEX and OPEX associated with owning and operating DCs and to ensure high scalability in DCs. The RSD reference model provides holistic guidelines to support the creation of go-to-market products by Intel RSD adapters. Examples of notable partners (original equipment manufacturers and telecom equipment manufacturers) that have adopted the Intel RSD reference model in commercial products include Super Micro Computers, Ericsson and DELL [53]. A multi-tier (electrical) Ethernet based fabric, which adopts silicon photonics and optical links, is proposed for interconnecting disaggregated resources in Intel's reference model. Ethernet is the preferred protocol because it is a versatile and low-cost networking technology.

A similar rack-scale composable infrastructure that adopts an electrical network topology and deploys optical links was proposed by Huawei in [86]. A pooled resource access protocol is implemented on bespoke electrical top of rack (ToR) switches and on cloud controllers attached to each resource node within the rack. Gen-Z [82], an industry led consortium, intends to design a new computing architecture that supports the disaggregation of processing and memory modules. Gen-Z has proposed a universal interface that interconnects disaggregated processing and memory modules to support access to remote byte addressable memory modules. Such interface enables the abstraction of the underlying memory technology; hence, diverse memory classes (i.e. legacy DRAM or novel SCM) may be attached to processing modules in Gen-Z compliant computing infrastructure. Gen-Z supports the interconnection of disaggregated compute modules via both direct-attached and switched fabric approaches. The consortium has also proposed high bandwidth and low (sub-100ns) latency electrical switches capable of interconnecting separated CPU and memory components over a switching fabric [90]. Such electrical switches can mitigate performance degradation of latency sensitive applications in a composable infrastructure. In other literature [62], [91], the adoption of electrical switches such as PCIe and InfiniBand switches are proposed for DC infrastructures with composable I/O, storage, and accelerators.

The disaggregated Recursive Datacentre-in-a-Box (dReDBox) was proposed in [61]. dReDBox deploys aggregated low power system on chips

(SoCs) in compute modules and aggregated memory components in memory modules within racks in the DC. Compute modules also contain DRAM which function as main memory; hence, partial disaggregation is employed. A three-tier network topology comprising of optical switches was proposed for the dRedBox in [92]. The primary switching technique adopted in network topology is optical circuit switching to achieve ultra-low latency communication. However, programmable chips (attached to each resource module) and the electronic switch (attached to each middle tier switch) also enable electronic packet/circuit switching in the topology. Hence, the topology adopts a distributed switching architecture because programmable chips in each resource module perform network switching functions.

Different variants of a three-tier network topology were proposed for composable DCs by the authors of [87]–[89]. The proposed topologies in [89], [93] implement a distributed switching architecture. This is because of field-programmable gate array (FPGA) based switch interface cards (SICs) that are adopted to replace the traditional NICs in each compute or memory node. The SICs also enable the creation of a full mesh connectivity between resource nodes in the same rack for low latency communication and perform network routing functions. However, the number of interfaces and links required to implement such intra-rack connectivity is about a square of the number of resource nodes in the rack. The authors in [78], adopted a 3D torus topology to interconnect micro-servers (resource nodes) within the same rack. This is because the 3D torus topology requires a small number of links per node. Each micro-server comprises of compute, memory, and network interfaces. A distributed switching architecture is employed. This is because each micro-server has an embedded switch fabric that forwards traffic between resource nodes in the same rack over a direct point-to-point physical topology.

In [14] a hybrid network topology is proposed for a rack-scale composable DC infrastructure. An optical switch in each rack is the fast-optical backplane that supports intra-rack communication between compute and remote memory nodes. A second (generic) backplane that begins in the rack and extends into a hybrid leaf-spine network (with both electrical and optical spine switches). The generic backplane conveys VM to VM, CPU-IO, RAM-IO, CPU-disk and RAM-disk traffics of the composable infrastructure. In [94], the authors also proposed an hybrid network topology for pod-scale composable infrastructure. The topology adopted two tiers of switches in each rack. The lower tier in each rack comprise of electronic switches that connect directly to each resource module in the rack via optical fibre links. Electronic

switches in the lower tier connect to non-blocking optical cross connect (OXC) which functions as the ToR switch. The OXC also provides full mesh connectivity between adjacent racks. In [95], the SDN-capable two-tiered hybrid network topology developed for conventional DCs is also expected to be applicable in disaggregated DCs. Table 3.4 gives a summary of some (research) projects on composable DCs in literature.

Software control is critical in composable DC; hence, SDI is a de facto feature of most composable infrastructure commercial solutions and prototypes [84], [86], [96]. Additionally, studies have also been conducted to demonstrate the efficacy of composable DCs and to derive policies or algorithms that will lead to maximum efficiency when deployed in composable DC. In [97], an orchestration platform that enables maximum resource utilisation in the dReDBox was described. Different algorithms for workload demand placement were demonstrated over the orchestration platform. The authors in [98] demonstrated that a modern DC scheduler is able to achieve better performance when deployed in a disaggregated DC relative to a traditional DC. Evaluation metrics used for the demonstration include average number of provisioned resource demands and average number of unused resource capacity. The work in [60] formulated an integer linear programming model and developed a simulated annealing based algorithm to optimally allocate virtual DC requests in a rack-scale DC. The optical DC interconnect proposed in [88] was adopted for the study. The model compared the benefits of disaggregated DCs to traditional DCs using VM rejection and resource utilisation as the primary metrics. Other metrics, such as power consumption of DC network, were not studied. The authors in [99] developed an energy efficient MILP model which optimally placed VMs in a pod-scale composable DC. The authors subsequently introduced an energy efficient heuristic in [100] as an extension of the initial work to show the advantages of disaggregation over traditional DC architecture.

Although, physical disaggregation at rack-scale and pod-scale have been widely studied, comparison of both approaches of disaggregation from the energy efficiency perspective is absence in the literature. Additionally, comparison of composable DC networks from an energy efficiency perspective is also missing. Furthermore, the concept of logical disaggregation has not been well explored. Neither has it been compared to other forms of disaggregation from an energy efficiency perspective. This thesis conducts a comparison of physically and logically disaggregated composable DCs over electrical, hybrid and optical DCNs from the energy

efficiency perspective. We also explore the adoption of the integrated workload architecture, which increases workload modularity, to improve overall energy efficiency in composable DCs.

In spite of the common view that the adoption of optical communication and silicon photonics is essential for composable DCs, optimal adoption such technologies remain an open issue. In this thesis, propose a novel network for composable DCs that optimally uses optical communication and silicon photonics in conjunction with electrical switches. Most application of disaggregation in literature are often focused on massive DCs. In contrast, this thesis, for the first time, considers the impact of adopting disaggregation for improved energy efficiency in a group of networked fog sites at the network edge.

Table 3.4: Composable DC research projects in literature

Project	**Disaggregation Scope**			**Disaggregation Type**		**Network Type**			**Network Architecture**	
	Physical	**Logical**	**Hybrid**	**Partial**	**Full**	**Electrical**	**Optical**	**Hybrid**	**Centralised**	**Distributed**
[83]	-	-	-	✓	-	-	-	-	-	-
[84]	-	-	-	✓	-	-	-	-	-	-
Intel RSD [81][85]	✓	-	-	✓	-	✓	-	-	✓	-
HTC-DC [86]	-	-	-	-	-	✓	-	-	-	-
Gen-Z [82]	-	-	-	-	-	✓	-	-	-	-
dReDBox [92][61]	✓	✓	✓	✓	-	-	-	✓	-	✓
DORIOS [87]	✓	-	-	✓	-	-	✓	-	-	✓
EVROS [88]	✓	-	-	-	✓	-	✓	-	-	✓
[89]	✓	-	-	-	✓	-	-	✓	-	✓
[78]	✓	-	-	✓	-	✓	-	-	-	✓
[14]	✓	-	-	✓	-	-	-	✓	-	-
[94]	✓	-	-	-	✓	-	-	✓	✓	-
NEPHELE [95]	-	-	-	-	-	-	-	✓	✓	-

3.7 Summary

This chapter presented an overview of composable DCs and supporting technologies. We also discussed the importance and role of supporting technologies and concepts of composable DC which include resource disaggregation, SDI and optical communication and silicon photonics. Furthermore, composable DC can enable increased modularity, greater agility and flexibility and improved efficiencies of computing infrastructure. This was highlighted. We also discussed network, SDI, and other implementation challenges of associated with composable DCs. We gave an extensive review of industry-led research and academic research which have been conducted to demonstrate the efficacy or to address mentioned challenges of composable DC. This thesis focuses on the energy efficiency of composable DCs by investigating the optimal scale and scope for resource disaggregation in composable DCs while considering compute and network energy efficiencies. Furthermore, two variants of a novel, simple, and practical network for composable DCs are proposed. Finally, we also conduct a pioneering investigation of the benefits and impact of employing composable servers for greater energy efficiency in federation of distributed fog computing sites at the network edges.

Chapter 4 : Energy Efficient Placement of Workloads in Composable Data Centres Networks

4.1 Introduction

The number and importance of date centres (DCs) continues to grow as the uptake of digital technologies spans various sectors of the society. Adoption of composable DCs, which employ resource disaggregation along with suitable networks, is expected to improve DC efficiency in different domains. In this chapter, we compare different scales of resource disaggregation in composable DC infrastructures via the formulation of a mixed integer linear programming (MILP) model. The MILP performs energy efficient placement of both monolithic and integrated workloads over selected electrical, optical and hybrid networks deployed in composable DCs. Finally, we develop a heuristic for energy efficient placement of workloads in composable DCs which replicates the trends produced by the MILP model.

4.2 A Review of Reference Network Topologies

A few network topologies have been proposed in existing literature for composable DC infrastructure adopting both rack-scale and pod-scale disaggregation as reviewed in Section 3.6. These network topologies adopt electrical, hybrid (electro-optic) and optical switching components and links. A study of energy efficient placement of workloads in composable infrastructure is conducted over certain reference network topologies from literature. One electrical, one hybrid and one optical network topologies are selected as reference network architecture for the purpose of this study. Intel's multi-tiered electrical network topology proposed for the Rack Scale Design (RSD) is the adopted electrical network topology. EVROS is the reference optical network topology adopted while the hybrid network topology proposed in [94] is also adopted as a reference network topology. These reference topologies are adapted to support traditional, rack-scale and pod-scale composable DCs as required.

4.2.1 Intel's RSD Electrical Network Topology

Intel proposed a multi-tiered Ethernet switching topology to interconnect partially disaggregated resources in the Intel RSD reference model [85]. High versatility and low cost of Ethernet switches (due to their wide deployment) are the primary motivation given for their adoption in the network topology. As shown in Figure 4.1, each homogenously resourced node in each rack of the

reference model connects to a top of rack (ToR) Ethernet switch via a 10 GbE Network Interface Card (NIC) on the node. ToR switches in each rack of a pod connect to an aggregation (DC) switch. The aggregation switch provides connectivity between intra-pod and inter-pod ToR switches. Aggregation switches also provide connectivity to external networks. Optionally, the Intel RSD supports a lower network tier. In the lower tier, drawer Ethernet switches within the rack act as intermediate switches between resource nodes and the ToR switch within a rack. Intel RSD supports optical links at all tiers of the proposed network topology.

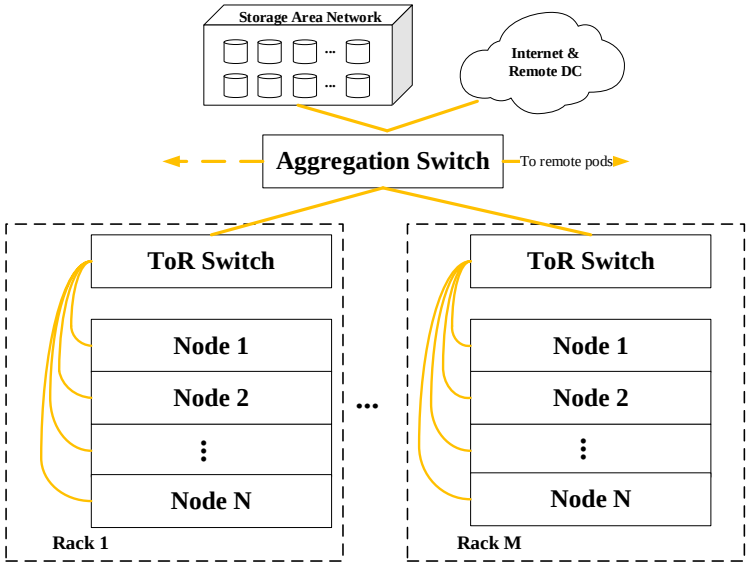


Figure 4.1: Electrical network topology for Intel's RSD

4.2.2 EVROS Optical Network Topology

The EVROS optical network topology comprises of inter-rack and intra-rack network topologies as shown in Figure 4.2. The inter-rack network topology adopts wavelength selective switches (WSSs) and optical circuit switches (OCS) along intra-rack and inter-rack communication paths. WSS perform the function of a ToR switch by inter-connecting resource nodes within the same rack and providing links to other WSSs in adjacent racks. Each ToR switch also connects to a top of cluster (ToC) WSS switch at the higher tier. The ToC switch in turn connects to an optical fibre switch.

As an alternative to the inter-rack network, the intra-rack network topology provides direct path between co-rack nodes. It is an intra-rack backplane that provides full mesh connectivity between intra-rack resource nodes via multi-port bespoke NICs called switch interface cards (SICs). A SIC is a multi-functional programmable field-programmable gate array (FPGA)-based component present in each resource node. SICs provide interfaces that

connect to both inter-rack and intra-rack networks. In addition, SICs perform the following functions:-

i. Read/write of data between local in-CPU memory and remote DRAM chip;
ii. Dynamic routing of data traffic from source to destination resource nodes via the intra-rack network, the inter-rack network or a combination of both networks; and
iii. Traffic aggregation using time division multiplexing mechanism for dynamic-sized frames transmission on demand.

EVROS also adopts low propagation delay hollow-core photonic bandgap fibre [101] for all links in the network topology.

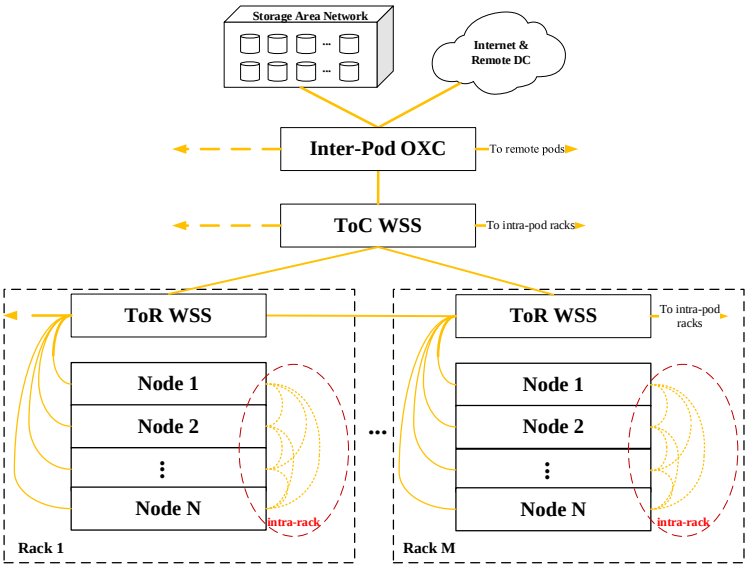


Figure 4.2: EVROS optical network topology

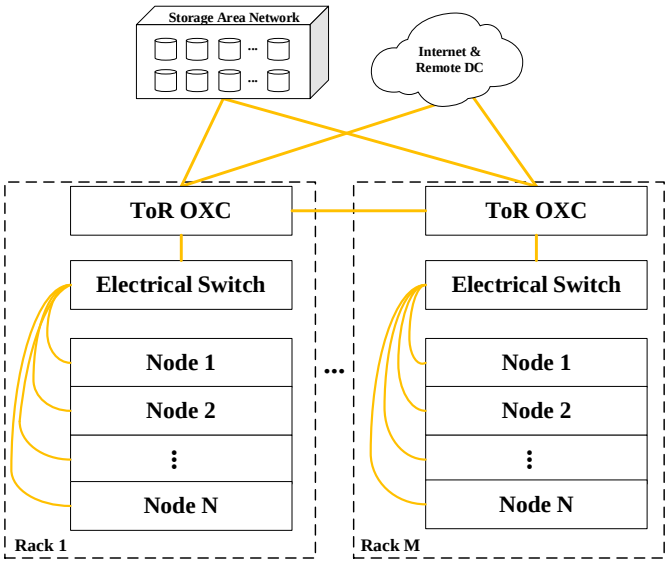


Figure 4.3: Hybrid network topology

4.2.3 Hybrid Network Topology

A hybrid network topology was proposed for pod-scale composable DCs in [94] . This network topology deploys two tiers of switches in each homogenously resourced rack as illustrated in Figure 4.3. The first tier comprises of electrical switches, which directly connects to each intra-rack resource node via optical links. The switches of the first tier also interface with ToR non-blocking optical switches, which provide full mesh connectivity between adjacent racks.

4.3 Energy Efficient Workloads Placement

4.3.1 Infrastructure Setup

The reference composable DC infrastructures are setup as follows; a DC comprises of one or more pods; each pod comprises of one or more racks and each rack comprises of one or more resource nodes. When physical disaggregation is implemented at pod-scale, each rack in a pod-scale DC comprise of one type (i.e. CPU or memory) of resource only. Hence, a logical server can only be formed at the pod-level. On the other hand, when physical resource disaggregation is implemented at rack-scale, each node placed in a rack of the rack-scale DC comprise of homogenous resources. However, each node in the rack can hold different resource types. Therefore, a logical server can be formed at the rack-level. Each rack of a traditional DC comprises of heterogeneous nodes, each node holds CPU and memory resource components. Consequently, a logical server can be formed within a node in the rack.

It is important to note that a memory resource component is used in two different contexts in the infrastructure setup. In the first, a memory resource component performs the function of RAM as defined in conventional computer architecture. In another context, a memory resource component also performs the function of a storage device. This is because in-memory computing is assumed for groups of workloads deployed in the DC. It is assumed that such groups of workloads perform in-memory data shuffle via remote direct memory access (RDMA).This reduces the impact of such data exchange on the CPU and the operating system. Hence, inter-memory traffic is created in the network based on the placement of the memory resource demand of workloads in the DC.

4.3.2 MILP Model Description

A MILP model is developed to minimise the total power consumption of composable DC infrastructures which employ the reference electrical, hybrid or optical network topology. Given resource allocation in composable DC infrastructures and workload templates generated by an orchestrator sitting above the physical infrastructure layer, the MILP model perform workload placement. The MILP model selects the optimum locations for each type of resource demanded by each workload so that the total DC power consumption is minimised.

In the modelled DC infrastructure, poor utilisation of direct attached storage is mitigated by centralising storage devices in remote systems such as a storage area network (SAN). Furthermore, data is retrieved over a unified network topology that supports both local area network traffic and SAN traffic via a bespoke offload NIC. Consequently, CPU or memory traffic to and from IO comprises of both SAN traffic and north-south traffic in the DC. IO traffic in the northbound direction always originates from the inter-DC switch interfaces. In the southbound direction, the inter-DC switch interfaces are always the destination of all IO traffic in the DC. At the inter-DC switch such traffic is relayed to the compute nodes, to the Internet or to the independent SAN system accordingly. CPU resource components used in the model have sufficient local cache to support remote memory access after compute disaggregation. Additionally, un-capacitated network state is assumed for all reference network topology to ensure fair comparison. This also allows further simplification of the model as it can be assumed that network traffic is routed through the shortest path. This is the expected best-case behaviour for each reference network topology. The sets, parameters and variables of the MILP model are introduced as follows.

Sets:

N	Set of nodes of resources
R	Set of racks in the DC
P	Set of pods in the DC
C	Set of CPU resources
M	Set of memory resources
W	Set of workloads
D	Set of traffic direction

DC Compute Parameters:

$\mathbb{C}_j$	Capacity of CPU module $j \in C$
CP_j	Maximum power consumption of CPU module $j \in C$
IC	Idle power as a fraction of maximum CPU power
ΔC_j	Power factor of CPU module $j \in C$; $\Delta C_j = \frac{CP_j - IC \cdot CP_j}{\mathbb{C}_j}$
$\mathbb{M}_j$	Capacity of memory module $j \in M$
MP_j	Maximum power consumption of memory module $j \in M$
IM	Idle power as a fraction of maximum memory power
ΔM_j	Power factor of memory module $j \in M$; $\Delta M_j = \frac{MP_j - IM \cdot MP_j}{\mathbb{M}_j}$
CN_{jn}	$CN_{jn} = 1$, If CPU $j \in C$ is placed in node $n \in N$. Otherwise $CN_{jn} = 0$
MN_{jn}	$MN_{jn} = 1$ if RAM $j \in M$ is placed in node $n \in N$, Otherwise $MN_{jn} = 0$
NR_{nr}	$NR_{nr} = 1$, If node $n \in N$ is placed in rack $r \in R$, otherwise $NR_{nr} = 0$
RP_{rp}	$RP_{rp} = 1$, If rack $r \in R$ is placed in pod $p \in P$, otherwise $RP_{rp} = 0$
WC_w	CPU capacity required by workload $w \in W$
WM_w	Memory capacity required by workload $w \in W$
$\mathbb{L}_w$	Maximum latency supported by DC infrastructure for workload $w \in W$
$\mathcal{T}_{cm}$	CPU-Memory latency between CPU component $c \in C$ and memory component $m \in M$. Inter-component latency.
$\mathcal{Q}$	A big number (100000)
$\mathcal{G}$	A big number (1000)
α	A weighing factor in Watt which specifies the cost per blocked workload

DC Network Parameters:

$\mathcal{X}$	Static power consumption of optical cross-connect (W)
$\mathcal{W}$	Static power consumption of WSS-based TOR switch (W)

$\mathbb{E}$	Load proportionate energy of electrical switch (J/b)
$\mathcal{E}$	Idle power consumption of electrical switch (W)
$\mathbb{a}$	Number of aggregation switches, $\mathbb{a} \geq 1$; $\frac{\mathbb{a}}{\mathbb{r}}$ is a fixed aggregation ratio.
$\mathbb{b}$	Number of inter-pod cross connects, $\mathbb{b} \geq 1$; $\frac{\mathbb{b}}{\mathbb{p}}$ is a fixed ratio.
$\mathcal{C}_{wx}$	CPU-Memory (RAM) traffic (in b/s) of workload $w \in W$ in direction $x \in D$.
$\mathcal{I}_{wx}$	CPU-IO traffic of workload $w \in W$ in direction $x \in D$.
$\mathcal{R}_{wx}$	Memory(storage)-IO traffic of workload $w \in W$ in direction $x \in D$.
$\mathcal{M}_{sd}$	Inter-memory(storage) traffic between source workload $s \in W$ and destination workload $d \in W$.
EU_{sd}	Electrical network load proportional energy per bit (J/b) due to traffic from CPU component $s \in C$ to memory (RAM) component $d \in M$.
ED_{sd}	Electrical network load proportional energy per bit (J/b) due to traffic from memory (RAM) component $s \in M$ to CPU component $d \in C$.
EC_{sd}	Electrical network load proportional energy per bit (J/b) due to traffic between memory (storage) component $s \in M$ and memory (storage) component $d \in M$.
EN	Electrical network load proportional energy per bit (J/b) due to north-south traffic.
HU_{sd}	Hybrid network load proportional energy per bit (J/b) due to traffic from CPU component $s \in C$ to memory (RAM) component $d \in M$.
HD_{sd}	Hybrid network load proportional energy per bit (J/b) due to traffic from memory (RAM) component $s \in M$ to CPU component $d \in C$.
HC_{sd}	Hybrid network load proportional energy per bit (J/b) due to traffic between memory (storage) component $s \in M$ and memory (storage) component $d \in M$.

HN Hybrid network load proportional energy per bit (J/b) due to north-south traffic.

OU_{sd} Optical network load proportional energy per bit (J/b) due to traffic from CPU component $s \in C$ to memory (RAM) component $d \in M$.

OD_{sd} Optical network load proportional energy per bit (J/b) due to traffic from memory (RAM) component $s \in M$ to CPU component $d \in C$.

OC_{sd} Optical network load proportional energy per bit (J/b) due to traffic between memory (storage) component $s \in M$ and memory (storage) component $d \in M$.

ON Optical network load proportional energy per bit (J/b) due to north-south traffic.

Variables:

$\boldsymbol{c_{wj}}$ $\boldsymbol{c_{wj}} = 1$ indicates that processing requirements of workload $w \in W$ are served by CPU $j \in C$. Otherwise, $\boldsymbol{c_{wj}} = 0$

$\boldsymbol{m_{wj}}$ $\boldsymbol{m_{wj}} = 1$, indicates that memory (RAM) request of workload $w \in W$ is served by RAM $j \in M$. Otherwise, $\boldsymbol{m_{wj}} = 0$

$\mathbb{c}_{\boldsymbol{j}}$ $\mathbb{c}_{\boldsymbol{j}} = 1$, if CPU $j \in C$ is active. Otherwise, $\mathbb{c}_{\boldsymbol{j}} = 0$

$\mathbb{m}_{\boldsymbol{j}}$ $\mathbb{m}_{\boldsymbol{j}} = 1$, if RAM $j \in M$ is active. Otherwise, $\mathbb{m}_{\boldsymbol{j}} = 0$

$\boldsymbol{p_p}$ $\boldsymbol{p_p} = 1$, if pod $p \in P$ is active. Otherwise, $\boldsymbol{p_p} = 0$

$\boldsymbol{r_r}$ $\boldsymbol{r_r} = 1$, if rack $r \in R$ is active. Otherwise, $\boldsymbol{r_r} = 0$

$\mathbb{w}_{\boldsymbol{w}}$ Indicates the state of workload $w \in W$ i.e., served or unserved.$\mathbb{w}_{\boldsymbol{w}} = 1$, if workload w is served. Otherwise, $\mathbb{w}_{\boldsymbol{w}} = 0$

$\boldsymbol{\beta_w}$ Indicates the state of workload $w \in W$ i.e., rejected or active. It is the opposite of $\mathbb{w}_{\boldsymbol{w}}$, $\boldsymbol{\beta_w} = 1 - \mathbb{w}_{\boldsymbol{w}}$

$\mathbb{r}$ Number of active racks

$\mathbb{p}$ Number of active pods

$\boldsymbol{h_{wr}}$ $\boldsymbol{h_{wr}} = 1$, if CPU resource demand of workload $w \in W$ is placed in rack $r \in R$. Otherwise, $\boldsymbol{h_{wr}} = 0$

$\boldsymbol{g_{wr}}$ $\boldsymbol{g_{wr}} = 1$, if memory resource demand of workload $w \in W$ is placed in rack $r \in R$. Otherwise, $\boldsymbol{g_{wr}} = 0$

$\boldsymbol{a_{wp}}$ $\boldsymbol{a_{wp}} = 1$, if CPU resource demand of workload $w \in W$ is placed in pod $p \in P$. Otherwise, $\boldsymbol{a_{wp}} = 0$

$\boldsymbol{b_{wp}}$ $\boldsymbol{b_{wp}} = 1$, if memory resource demand of workload $w \in W$ is placed in pod $p \in P$. Otherwise, $\boldsymbol{b_{wp}} = 0$

$\boldsymbol{y_{wcm}}$ Indicates the CPU-memory pair used to provision workload $w \in W$.$\boldsymbol{y_{wcm}}$= 1 if CPU $c \in C$ and memory $m \in M$ host CPU and memory resource demands of workload $w \in W$ respectively. Otherwise, $\boldsymbol{y_{wcm}}$ = 0.

$\boldsymbol{z_{sd}^{xy}}$ $\boldsymbol{z_{sd}^{xy}} = 1$ if memory resource demand of source workload $s \in W$ is placed in memory component $x \in M$ and memory resource demand of destination workload $d \in W$ is placed in memory component $y \in M$. Otherwise, $\boldsymbol{z_{sd}^{xy}} = 0$

The creation of the CPU requirements of workload w, $\mathbb{w}_w$, can be related to the workload placement using:

$$\mathbb{w}_w = \sum_{j \in CR} \boldsymbol{c_{wj}} \qquad (4.1)$$

$$\forall\, w \in W$$

The number of active racks is the DC can be derived from the state of each rack using:

$$\mathbb{r} = \sum_{r \in R} \boldsymbol{r_r} \qquad (4.2)$$

The number of active pods is the DC can be derived from the state of each pod using:

$$\mathbb{p} = \sum_{p \in P} \boldsymbol{p_p} \qquad (4.3)$$

The placement of the CPU resource requirement of workload w into rack r can be derived using:

$$\boldsymbol{h_{wr}} = \sum_{n \in N} \sum_{j \in C} \boldsymbol{c_{wj}} CN_{jn} NR_{nr} \qquad (4.4)$$

$$\forall\, w \in W, \forall\, r \in R$$

The placement of the memory resource requirement of workload w into rack r can be derived using:

$$\boldsymbol{g_{wr}} = \sum_{n \in N} \sum_{j \in M} \boldsymbol{m_{wj}} MN_{jn} NR_{nr} \qquad (4.5)$$

$$\forall w \in W, \forall r \in R$$

The placement of the CPU resource requirement of workload w into pod p can be derived using:

$$\boldsymbol{a_{wp}} = \sum_{r \in R} \sum_{n \in N} \sum_{j \in C} \boldsymbol{c_{wj}} CN_{jn} NR_{nr} RP_{rp} \tag{4.6}$$

$$\forall w \in W, \forall p \in P$$

The placement of the memory resource requirement of workload w into pod p can be derived using:

$$\boldsymbol{b_{wp}} = \sum_{r \in R} \sum_{n \in N} \sum_{j \in M} \boldsymbol{m_{wj}} MN_{jn} NR_{nr} RP_{rp} \tag{4.7}$$

$$\forall w \in W, \forall p \in P$$

Given each network topology, the load proportional power consumption of network components traversed and some variables, the total network power consumption (TNPC) of electrical, hybrid and optical network topologies are derived as follows:

Electrical network Topology

$$\begin{aligned} TNPC = & \sum_{c \in C} \sum_{m \in M} \sum_{w \in W} \boldsymbol{y_{wcm}} (\mathcal{C}_{w1}\, EU_{cm} + \mathcal{C}_{w2}\, ED_{cm}) \\ & + \sum_{x \in M} \sum_{y \in M:\, x \neq y} \sum_{s \in W} \sum_{d \in W: s \neq d} \boldsymbol{z_{sd}^{xy}} \mathcal{M}_{sd}\, EC_{xy} \\ & + EN \left(\sum_{w \in W} \sum_{c \in C} \boldsymbol{c_{wj}} (\mathcal{I}_{w1} + \mathcal{I}_{w2}) \right. \\ & \left. + \sum_{w \in W} \sum_{m \in M} \boldsymbol{m_{wj}} (\mathcal{R}_{w1} + \mathcal{R}_{w2}) \right) + (\mathbb{r} + \mathbb{a})\mathcal{E} \end{aligned} \tag{4.8}$$

Hybrid Network Topology

$$\begin{aligned} TNPC = & \sum_{c \in C} \sum_{m \in M} \sum_{w \in W} \boldsymbol{y_{wcm}} (\mathcal{C}_{w1}\, HU_{cm} + \mathcal{C}_{w2}\, HD_{cm}) \\ & + \sum_{x \in M} \sum_{y \in M:\, x \neq y} \sum_{s \in W} \sum_{d \in W: s \neq d} \boldsymbol{z_{sd}^{xy}} \mathcal{M}_{sd}\, HC_{xy} \\ & + HN \left(\sum_{w \in W} \sum_{c \in C} \boldsymbol{c_{wj}} (\mathcal{I}_{w1} + \mathcal{I}_{w2}) \right. \\ & \left. + \sum_{w \in W} \sum_{m \in M} \boldsymbol{m_{wj}} (\mathcal{R}_{w1} + \mathcal{R}_{w2}) \right) + \mathbb{r}(\mathcal{X} + \mathcal{E}) \end{aligned} \tag{4.9}$$

Optical Network Topology

$$TNPC = \sum_{c \in C} \sum_{m \in M} \sum_{w \in W} \boldsymbol{y_{wcm}} (\mathcal{C}_{w1}\, OU_{cm} + \mathcal{C}_{w2}\, OD_{cm}) + \sum_{x \in M} \sum_{y \in M:\, x \neq y} \sum_{s \in W} \sum_{d \in W: s \neq d} \boldsymbol{z_{sd}^{xy}} \mathcal{M}_{sd}\, OC_{xy} + ON \left(\sum_{w \in W} \sum_{c \in C} \boldsymbol{c_{wj}} (\mathcal{I}_{w1} + \mathcal{I}_{w2}) + \sum_{w \in W} \sum_{m \in M} \boldsymbol{m_{wj}} (\mathcal{R}_{w1} + \mathcal{R}_{w2}) \right) + \mathbb{r}\, \mathcal{W} + (\mathbb{p} + \mathbb{b}) \mathcal{X} \tag{4.10}$$

Total CPU Power Consumption

Total power consumption of CPU resources in the composable DC ($TCPC$) is derived as follows.

$$TCPC = \sum_{j \in C} \sum_{w \in W} \left((\mathrm{IC}\, CP_j \mathbb{c}_{\boldsymbol{j}}) + (\Delta C_j\, \boldsymbol{c_{wj}}\, WC_w) \right) \tag{4.11}$$

Total Memory Power consumption

Total power consumption of memory resources in the composable DC ($TMPC$) is derived as follows.

$$TMPC = \sum_{j \in M} \sum_{w \in W} \left((\mathrm{IM}\, MP_j \mathbb{m}_{\boldsymbol{j}}) + (\Delta M_j\, \boldsymbol{m_{wj}}\, WM_w) \right) \tag{4.12}$$

The MILP model is defined as follows:

Objective: Minimise

$$TCPC + TMPC + TNPC + \alpha \left(\sum_{w \in W} \boldsymbol{\beta_w} \right) \tag{4.13}$$

Equation (4.13) is the objective of the model; it minimises total power consumption of CPU resources, memory resources, and of the network topology used. It also minimises the number of rejected workloads in scenarios where some workloads cannot be provisioned. α is the cost (measured in Watt) associated with each rejected workload.

Subject to:

$$\sum_{w \in W} WC_w\, \boldsymbol{c_{wj}} \leq \mathbb{C}_j \quad \forall\, j \in C \tag{4.14}$$

$$\sum_{w \in W} WM_w\, \boldsymbol{m_{wj}} \leq \mathbb{M}_j \quad \forall\, j \in M \tag{4.15}$$

Constraints (4.14) and (4.15) denote resource capacity constraints for each unit of CPU and memory component in the DC.

$$\sum_{j \in C} \boldsymbol{c_{wj}} \leq 1 \tag{4.16}$$

$$\forall w \in W$$

$$\sum_{j \in M} \boldsymbol{m_{wj}} \leq 1 \tag{4.17}$$

$$\forall w \in W$$

Constraints (4.16) and (4.17) limit the maximum number of components that can host CPU and memory resource requests of a workload to one. This is because neither replication nor slicing of workloads is permitted. These constraints also allow workloads to be rejected in scenarios where resource capacity is limited.

$$\sum_{j \in C} \boldsymbol{c_{wj}} = \sum_{j \in M} \boldsymbol{m_{wj}} \tag{4.18}$$

$$\forall w \in W$$

Constraint (4.18) ensures that an active workload's CPU and memory resource requirements are satisfied. Otherwise the workload is inactive.

$$\mathcal{G} \sum_{w \in W} \boldsymbol{c_{wj}} \geq \mathbb{c}_{\boldsymbol{j}} \tag{4.19}$$

$$\forall j \in C$$

$$\sum_{w \in W} \boldsymbol{c_{wj}} \leq \mathcal{Q}\, \mathbb{c}_{\boldsymbol{j}} \tag{4.20}$$

$$\forall j \in C$$

$$\mathcal{G} \sum_{w \in W} \boldsymbol{m_{wj}} \geq \mathbb{m}_{\boldsymbol{j}} \tag{4.21}$$

$$\forall j \in M$$

$$\sum_{w \in W} \boldsymbol{m_{wj}} \leq \mathcal{Q}\, \mathbb{m}_{\boldsymbol{j}} \tag{4.22}$$

$$\forall j \in M$$

Constraints (4.19) - (4.22) determine each CPU and memory resource state, this depends on utilisation of the resource to satisfy resource requirements of served workloads.

$$\sum_{w \in W} (\boldsymbol{h_{wr}} + \boldsymbol{g_{wr}}) \geq \mathbb{r}_r \tag{4.23}$$

$$\forall\, r \in R$$

$$\sum_{w \in W} (\boldsymbol{h_{wr}} + \boldsymbol{g_{wr}}) \leq \mathcal{Q}\, \mathbb{r}_r \tag{4.24}$$

$$\forall\, r \in R$$

Constraints (4.23) and (4.24) determine the state of each rack, this depends on the utilisation of CPU or memory resource in the rack to satisfy resource requirements of served workloads.

$$\mathcal{G} \sum_{w \in W} (\boldsymbol{a_{wp}} + \boldsymbol{b_{wp}}) \geq \mathbb{p}_p \tag{4.25}$$

$$\forall\, p \in P$$

$$\sum_{w \in W} (\boldsymbol{a_{wp}} + \boldsymbol{b_{wp}}) \leq \mathcal{Q}\, \mathbb{p}_p \tag{4.26}$$

$$\forall\, p \in P$$

Constraints (4.25) and (4.26) determine the state of each pod, this depends on the utilisation of CPU or memory resource in the pod to satisfy resource requirements of served workloads.

$$\boldsymbol{y_{wcm}} = \boldsymbol{c_{wc}}\, \boldsymbol{m_{wm}} \tag{4.27}$$

$$\forall\, w \in W, \forall\, c \in C, \forall m \in M$$

Constraint (4.27) gives the relationship between components hosting processing and memory resource demands of a given workload i.e., $\boldsymbol{y_{wcm}}$ which is a product of $\boldsymbol{c_{wc}}$ and $\boldsymbol{m_{wm}}$.

$$\boldsymbol{y_{wcm}} \leq \boldsymbol{c_{wc}} \tag{4.28}$$

$$\forall\, w \in W, \forall\, c \in C, \forall m \in M$$

$$\boldsymbol{y_{wcm}} \leq \boldsymbol{m_{wm}} \tag{4.29}$$

$$\forall\, w \in W, \forall\, c \in C, \forall\, m \in M$$

$$\boldsymbol{y_{wcm}} \geq \boldsymbol{c_{wc}} + \boldsymbol{m_{wm}} - 1 \tag{4.30}$$

$$\forall\, w \in W, \forall\, c \in C, \forall\, m \in M$$

Constraints (4.28) - (4.30) are used to linearise constraint (4.27), which is a product of two binary variables.

$$\sum_{c \in C} \sum_{m \in M} \mathcal{T}_{cm}\, \boldsymbol{y_{wcm}} \leq \mathbb{L}_w \qquad (4.31)$$

$$\forall\, w \in W$$

Constraint (4.31) ensures that the inter-component latency between the CPU and memory components hosting a workload's resource demands does not exceed the set maximum CPU-Memory latency for a given type of composable DC infrastructure. This constraint is combined with the allocation of components dictates the type of composable DC infrastructure under consideration and enforces resource locality. Component allocation is given by parameters $CN_{jn}, MN_{jn}, NR_{nr}$ and RP_{rp}.

$$\boldsymbol{z_{sd}^{xy}} = \boldsymbol{m_{sx} m_{dy}} \qquad (4.32)$$

$$\forall\, s \in W, \forall\, d \in W, \forall\, x \in M, \forall\, y \in M$$

$$\boldsymbol{z_{sd}^{xy}} \leq \boldsymbol{m_{sx}} \qquad (4.33)$$

$$\forall\, s \in W, \forall\, d \in W, \forall\, x \in M, \forall\, y \in M$$

$$\boldsymbol{z_{sd}^{xy}} \leq \boldsymbol{m_{dy}} \qquad (4.34)$$

$$\forall\, s \in W, \forall\, d \in W, \forall\, x \in M, \forall\, y \in M$$

$$\boldsymbol{z_{sd}^{xy}} \geq \boldsymbol{m_{sx}} + \boldsymbol{m_{dy}} - 1 \qquad (4.35)$$

$$\forall\, s \in W, \forall\, d \in W, \forall\, x \in M, \forall\, y \in M$$

Constraint (4.32) gives the relationship between two memory components ($x \in M$ and $y \in M$) hosting memory resource demands of workloads ($s \in W$ and $d \in W$) respectively i.e., $\boldsymbol{z_{sd}^{xy}}$ which is a product of binary variables $\boldsymbol{m_{sx}}$ and $\boldsymbol{m_{dy}}$. Constraints (4.33) - (4.35) are used to linearise constraint (4.32).

4.4 Performance Evaluation

The MILP model is used to study the performance of composable DCs that employ physical disaggregation at rack-scale and pod-scale relative to the performance of traditional DC infrastructure. Heterogeneous CPU and memory resource component are adopted to reflect heterogeneity of resources in production DCs. Three classes of servers illustrated in Table 4.1 are adopted to form a range of heterogeneous resourced DCs. To minimise the execution time of the MILP model which grows as the size and complexity of the problem increases, a small DC site comprising of 24 servers (i.e., 24 CPU and 24 memory components) is considered; 8 servers from each server class are selected. When the traditional DC is implemented at the site, servers maintain their single-box architecture. The implementation of rack-scale and

pod-scale DCs at the site requires the disaggregation of the components of servers accordingly.

Table 4.1: Server classification

Server class	**CPU capacity (Peak power)**	**RAM capacity (Peak power)**
High Performance (HPS)	3.6 GHz (130 W) [99]	32 GB (40 W)
Standard (STDS)	2.66 GHz (95 W) [99]	24 GB (30.72 W) [99]
Legacy (LS)	2.4 GHz (80 W) [99]	8 GB (10.24 W) [99]

Resources are allocated as follows:

- A common DC site is adopted under traditional, rack-scale and pod-scale DC scenarios considered in this chapter. The DC site has 2 pods, a pod comprises of two heterogeneously/homogenously resourced racks. Each rack holds multiple homogenously or heterogeneously resourced nodes. The DC site also comprises of 24 CPU and 24 memory components which are allocated to nodes within each rack.
- Each rack of the traditional DC is a heterogeneous rack that holds 6 heterogeneous nodes i.e., traditional servers (2 servers from each class of server defined). The traditional DC represents node-scale disaggregation and the performance of rack-scale and pod-scale DC are benchmarked against the traditional DC in this chapter.
- Each rack of rack-scale DC holds 3 homogeneous nodes of CPU resources and 3 homogenous nodes of memory resources. Hence, rack-scale DC comprises of heterogeneously resourced racks, 2 of these heterogeneous racks are allocated to each pod.
- In a pod-scale DC, racks hold 6 homogenous nodes of CPU or memory resources i.e., each rack comprises of homogenous nodes of CPU or memory resources. Each pod comprises of 1 homogeneous rack of CPU resources and 1 homogenous rack of memory resources.

Allocation of resource components in traditional, rack-scale, and pod-scale DCs determines the inter-component latency between any CPU and memory resource component pair in the DC. Table 4.2 gives the abstracted inter-component latency between pair DC components (i.e., CPU and memory) in traditional DCs and in a DC that is physically disaggregated DCs at rack-scale or pod-scale. The abstraction of latency values in Table 4.2 is obtained relative to the minimum physical separation required (between two resource components) to implement a desired composable DC. However, other metrics such as transmission latency may also be adopted to abstract

latency values. Resource component allocation and the choice of maximum latency ($\mathbb{L}_w$) for each workload in Constraint (4.31) collectively determine the composable DC infrastructure being evaluated by the MILP model. For simplicity, it is assumed that each traditional server has a single CPU component and a single memory component before disaggregation. Hence, both CPU and memory components of a given server can share a common index. Therefore, $\mathcal{T}_{cm}$ is derived from resource component allocation i.e., $CN_{jn}, MN_{jn}, NR_{nr}$ and RP_{rp} as described earlier. This derivation is guided by the latency abstraction in Table 4.2. Similarly, allocation of resource components in DCs also aids the pre-computation of load proportional energy consumption of network components traversed on each reference network topology considered i.e., EU_{sd}, ED_{sd}, EC_{sd}, EN, HU_{sd}, HD_{sd}, HC_{sd}, HN, OU_{sd}, OD_{sd}, OC_{sd}, and ON.

Table 4.2: Composable DC inter-component latency abstraction

DC Type	**Relative location of pair component**	**Representative values**
Traditional	Pair components in the same server/node.	1
Rack-scale	Pair components in different servers within the same rack.	2
Pod-scale	Pair components in different racks within the same pod.	3
DC-scale	Pair components in different pods within the same DC	4

In addition to the fixed idle power consumption, each CPU and memory resource component has a linear load proportionate power profile. The “power factor” which represents the slope of the linear power profile is the active power consumed per resource capacity. This is the basis for calculating the load proportionate power of each active CPU and memory components. However, the power factor of a resource component alone does not give a full picture of its energy efficiency. This is because a resource component with low power factor may have little capacity; hence, it can only support a small volume of resource demand. Normalising the power factor of components by their corresponding capacity gives a better measure of energy efficiency as shown in Figure 4.4. Therefore, at full (100%) CPU resource utilisation, high performance server (3.6 GHz) CPU is the most energy efficient, followed by

standard server (2.66 GHz) CPU. Thereafter, the legacy server (2.4 GHz) CPU closely follows standard server CPU as shown in Figure 4.4. Energy efficiency of fully utilised memory components also follow the same order. The idle power consumption of the CPU and memory resources is 70% of the maximum power. This is because a server in an idle state consumes up to 70% of its peak power[102]. Overall, from Table 4.1, CPU components peak power consumption is relatively higher than that of memory components. The peak power and idle power as a fraction of maximum power of both CPU and memory components are given in Table 4.1 and Table 4.3 respectively. Table 4.3 also gives the load proportionate energy per bit values of next-generation network interfaces (using silicon photonics technologies) at different tiers of DC infrastructures.

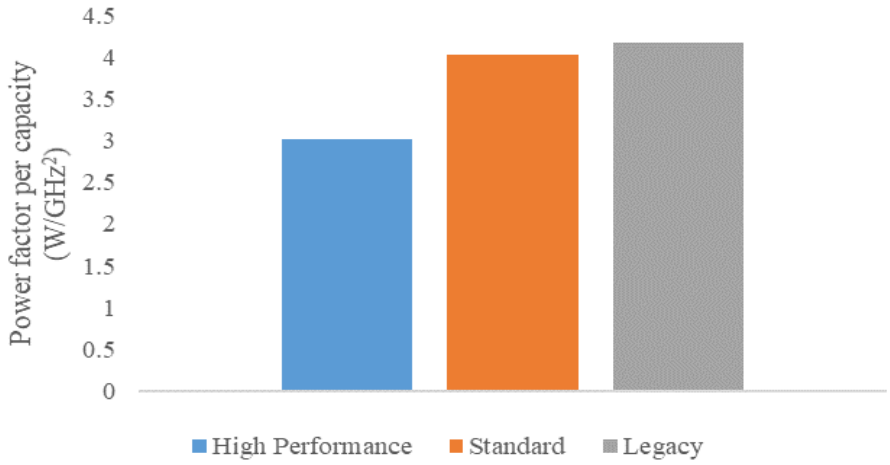


Figure 4.4: Power factor per capacity of server CPU components

The power consumption of electrical and optical components in different candidate network topologies are evaluated using the parameters given in Table 4.3. Because a small DC infrastructure setup is adopted, parameters a and b which represent the number of aggregation switches in each pod and the number of inter-pod cross connects (in corresponding network topologies) respectively are both set to one. The load proportionate energy (28.28 pJ/b) of electrical switches is the ratio of dynamic power range (181 W) of the switch to the maximum switching capacity (6.4 Tbps) of a Cisco Nexus 3132C-Z [103] electrical switch. The dynamic power range of a component is given by the difference of the maximum and idle power consumption of that component. Programmable Finisar 4x16 WSS [104] (with maximum power consumption of 50 W) is adopted as the WSS-based ToR switch in the optical network topology. An all-optical circuit switch [105] with maximum power consumption of 75 W is deployed as the OXC in both optical and hybrid network topologies. These values and the energy efficiency values of next generation silicon

photonics networks (adopted from [80]) are used as default input parameters. to the MILP model. Next-generation networks are expected to be more energy and cost efficient than today's networks; hence, the low values for next-generations networks as given in Table 4.3. Relative to communication between nodes in the same or different racks over longer distances, on-board communication between components in compute node requires lower power consumption. This is because on-board communication requires a simpler design. Compared to long-distance communication between distributed DCs over a DC interconnect, inter-node communication between nodes within the DC is more energy efficient. This is because shorter distances are travelled within the DC. Hence, less complex interfaces with lower power consumption are required for intra-DC networks relative to inter-DC networks.

Table 4.3: DC components and interfaces power consumption

Description	Value
Idle power as a fraction of maximum CPU power	70%
Idle power as a fraction of maximum memory power	70%
On-board network interface energy per bit (J/b)	0.5 pJ/b [80]
Inter-rack network interface energy per bit (J/b)	1 pJ/b [80]
Rack backplane network interface energy per bit (J/b)	1 pJ/b [80]
Inter-DC network interface energy per bit (J/b)	10 pJ/b [80]
Peak power consumption of optical circuit switch	75 W [105]
Peak power consumption of WSS-based optical switch	50 W [104]
Peak power consumption of electrical switch	493 W [103]
Typical operating (idle) power of electrical switch	312 W [103]
Load proportional energy of electrical switches	28.28 pJ/b

Heterogeneity of DC workloads is considered by adopting two classes of workloads i.e., CPU intensive and memory intensive workloads. The resource demand of monolithic workloads in each class is illustrated in Table 4.4. The corresponding number of workloads required for each evaluation scenario are selected serially from Table 4.4. It is assumed that cache coherent traffic is limited to each CPU and does not traverse the DC network fabric [106]. Furthermore, fixed inter-resource communication traffics for each workload is considered to ensure fair comparison between different composable DC

infrastructures. CPU-memory traffic of each workload in the forward and backward directions are 120 Gbps and 100 Gbps respectively. CPU-IO traffic or Memory-IO traffic of each workload in the forward and reverse directions are 2 Gbps and 1 Gbps respectively.

Table 4.4: Monolithic workloads resource demand

Workload ID	**CPU Intensive**		**Memory Intensive**	
	CPU demand (GHz)	**Memory demand (GB)**	**CPU demand (GHz)**	**Memory demand (GB)**
1	2.6	6.7	1.6	17.7
2	2.3	6.3	1.5	15.7
3	1.8	5.9	1	14.3
4	2	5.8	1.2	13.7
5	2.8	7	1.8	19.4
6	2.2	7.2	1.4	19.8
7	1.5	6.3	0.8	16
8	1.6	6.8	0.9	18.1
9	2.4	7.6	1.6	21.8
10	3	6.9	2	18.4
11	1.6	6.7	0.9	17.5
12	2.8	6.7	1.9	17.6
13	1.9	8	1.1	23.4
14	1.6	6.1	0.9	15
15	2.1	7	1.3	19
16	2	5.1	1.2	10.2
17	2.7	6.1	1.7	14.8
18	1.7	6.7	1	17.9
19	1.9	6.5	1.1	17.3
20	1.8	5.1	1.1	10.3

In most situation, it is assumed that inter-memory traffic is bandwidth intensive or non-bandwidth intensive over a specified range. Uniform distribution of non-intensive and intensive in-memory traffic i.e., $\mathcal{M}_{sd}$, between workloads is performed over 0 - 10 Gbps and 10 - 70 Gbps ranges respectively. Workloads are clustered into groups of five to represent groups of associated applications in conventional DCs such as worker and master nodes. Each workload group of associated workloads has one-to-one, one-to-many, many-to-many, or mixed inter-memory traffic patterns between the workloads in that group.

The described MILP model evaluates the impact of electrical, hybrid and optical network topologies on the performance of rack-scale or pod-scale composable DCs relative to traditional DC infrastructure. The MILP model is solved using the 64-bit AMPL/CPLEX solver on the ARC3 supercomputing node with 24 CPU cores and 128 GB of memory [70]. The MILP model results analysis consider metrics such as CPU, memory and network power consumption, number of active DC resources and average active resource utilisation. Average active DC resource utilisation represents the average utilisation of all active CPU or memory resource components in the DC. The average utilisation of network components such as switches is not considered. To obtain optimal results, the MILP model bin-packs workloads resource demands onto DC resources to achieve optimal resource power and utilisation efficiencies. This is performed within capacity and resource locality constraints. Consequent of workload placement, inter-resource communication (between CPU-memory component pairs for a workload) and inter-memory data shuffle traffic traverse different network topologies.

4.4.1 CPU Intensive Workloads

Relative to other DC types, the traditional DC infrastructure has the highest quantity of active DC resources (CPU and memory i.e., server) under all network topologies considered. Similarly, the traditional DC has the lowest average active memory utilisation relative to other DC infrastructures considered. The results obtained when 20 CPU intensive workload are optimally provisioned as illustrated in Figure 4.5 shows this. These observations are consistent with the widely reported challenges of provisioning monolithic workloads in traditional DCs. Traditional DCs are characterised by disproportionate utilisation of DC resource [61], [86]. The CPU intensive nature of input workloads is responsible for high average active CPU utilisation observed in traditional DC as shown in Figure 4.5.

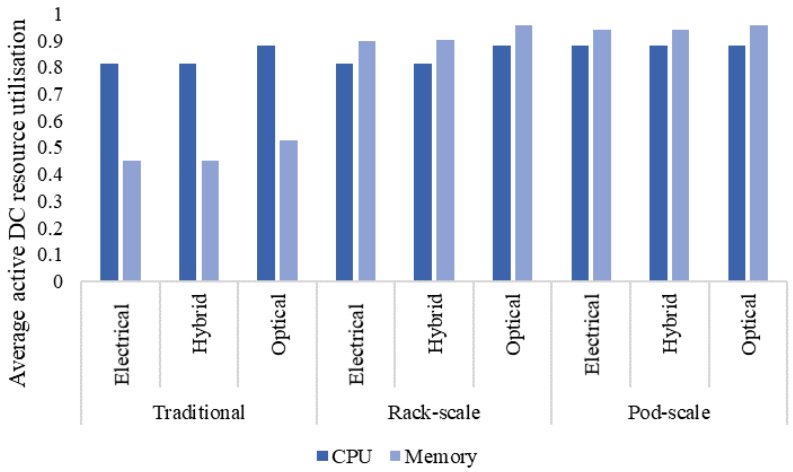


Figure 4.5: Average utilisation of active DC resource components under 20 CPU intensive workloads

For each number of CPU intensive workloads provisioned, the total CPU power consumption (TCPC) observed is equal for traditional DCs that employed electrical and hybrid network topologies. Figure 4.6 and Figure 4.7 show this. The total memory power consumption (TMPC) also follows the same trend under both electrical and hybrid network topologies. On the other hand, Figure 4.8 shows that the contributions of TCPC and/or TMPC to the total DC power consumption (TDPC) under the optical network topology often follow a different trend. Lower network power consumption per active rack in the optical network topology is responsible for this. The adoption of the optical network promotes the use of a different mix of active servers (CPU and memory resource components as seen in Figure 4.9 and Figure 4.10). The activated servers are selected from any rack in the DC if it leads to lower TCPC and TMPC. Consequently, relative to the hybrid or electrical networks, lower TDPC is often observed under the optical network.

In contrast, there is a preference for consolidation of workloads into a few active racks (over the use of servers distributed across racks in the DC) when electrical and hybrid network topologies are used. This leads to activation of few racks and ensures lower TDPC in traditional DCs and consequently leads to better energy efficiency. The presence of electrical switches (which have significant idle power consumption) in the lowest tiers of electrical and hybrid network topologies encourages this trend. For example, consider the scenario where 20 CPU intensive workloads are optimally provisioned in traditional DC. The TCPC and TMPC of traditional DC with optical network topology are 6% and 11% respectively lower than the values reported for traditional DC with electrical or hybrid network topologies. Figure 4.9 and Figure 4.10 show the disparity in server (CPU and memory)

resource usage under different network topologies when 20 CPU intensive workloads are provisioned.

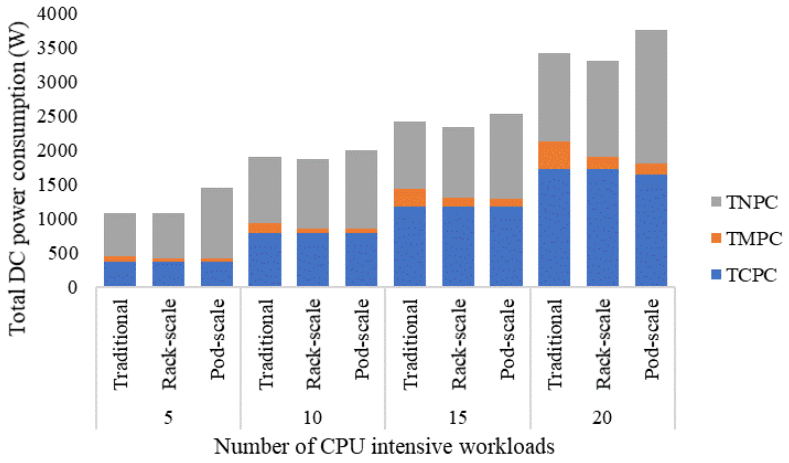


Figure 4.6: Total DC power consumption under CPU intensive workloads when electrical network topology is deployed

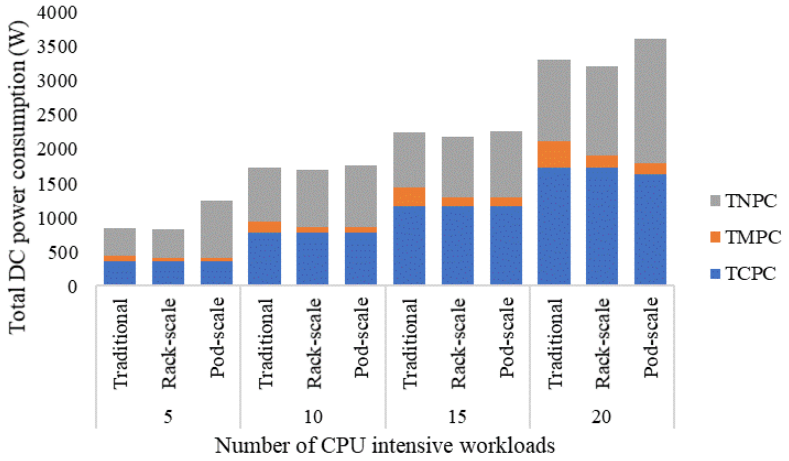


Figure 4.7: Total DC power consumption under CPU intensive workloads when hybrid network topology is deployed

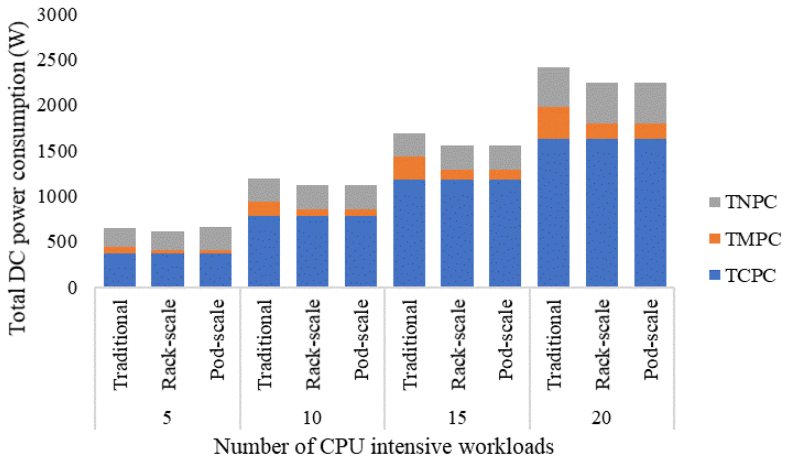


Figure 4.8: Total DC power consumption under CPU intensive workloads when optical network topology is deployed

For instance, Figure 4.9 shows that 3.6 GHz or 2.4 GHz CPUs are preferred over 2.66 GHz to achieve optimal TDPC when 20 CPU intensive workloads are provisioned in a traditional DC that deploys the optical network topology. On the one hand, this is because the 3.6 GHz CPU provides maximum energy efficiency when highly utilised as shown in Figure 4.4. On the other hand, the energy efficiency of 2.66 GHz and 2.4 GHz CPUs are somewhat comparable. Hence, a highly utilised 2.4 GHz CPU is more energy efficient than a partly utilised 2.66 GHz CPU. Therefore, the use of 3.6 GHz CPUs and their corresponding servers is promoted. This enables optimal consolidation of workload resource demands in a traditional DC that employs the energy efficient optical network. Furthermore, since network power consumption of the optical network is low, all eight 3.6 GHz CPU are activated to achieve optimal consolidation of workload resource demand. It is important to recall that each 3.6 GHz CPU is in a unique server and that servers are distributed across different racks in the traditional DC. Such consolidation leads to higher energy efficiency since fewer active servers are required. Once, all 3.6 GHz CPUs have been utilised for consolidation of workloads, 2.4 GHz CPUs are selected to provision the CPU resource demand of other workloads. These are workloads that are consolidated with other workloads. This is because the 2.4 GHz CPU is more energy efficient than the 2.66 GHz CPU at lower utilisation. 2.66 GHz CPU only becomes more energy efficient that the 2.4 GHz when it is highly utilised. This trend is discouraged when the electrical or hybrid network topology is deployed in the traditional DC due to the adoption of electrical switches in each rack. Hence, it is preferred to minimise the number of active racks. It is also important to note that the

selection of active resource components that leads to optimal TDPC is expected to change as the input workloads to the MILP model is revised.

The TNPC in traditional DC depends on the placement of workload resource demand into resource components. This is because workload resource demand placement determines the number of active racks and traffic that flows in the tiers of data centre network (DCN) topologies. As expected, TNPC changes with the network topologies as seen in Figure 4.6, Figure 4.7, and Figure 4.8. This is because of the variance in the power consumption profiles of components and interfaces that are present in the tiers of each network topology. In traditional DCs, high bandwidth inter-component traffics are node-limited i.e., they are restricted to on-board backplane of servers. On the other hand, inbound and outbound traffics of the DC (traffic between DC resources and remote systems) flow through higher tiers of each network topology. The contributions of node-limited traffic to the TNPC is small and constant across all network topologies in the traditional DCs. Relatively higher energy efficiency of on-board backplane compared to other tiers of each DCN topology (as illustrated in Table 4.3) is responsible for this. Although lower bandwidth traffic traverses the electrical network topology in a traditional DC, the electrical network topology has the highest TNPC. This is because of significant idle power consumption of electrical switches of the multi-tier topology. The TNPC of the hybrid network topology closely follows that of the electrical network topology. In addition to the fixed low power consumption of OXCs present in the higher tier of the hybrid topology, each rack in also has an electrical switch with significant idle power consumption. As shown in Figure 4.8, the optical network topology has the lowest TNPC. The fixed-low power consumption of poorly utilised high-capacity optical switches in the topology enabled this.

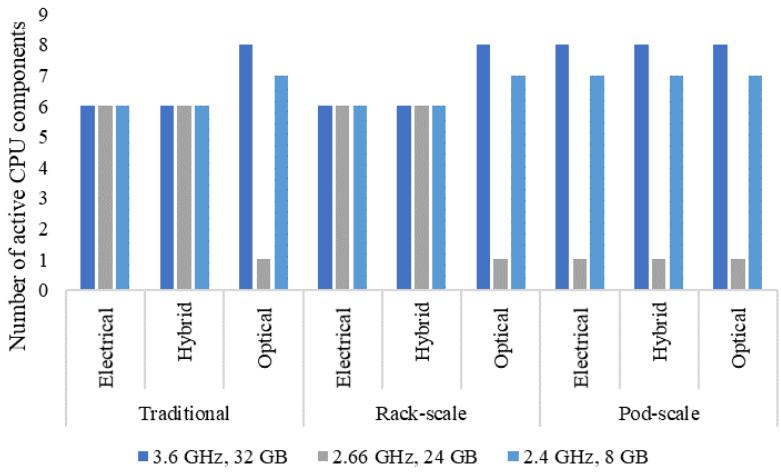


Figure 4.9: Active CPU components under 20 CPU intensive workloads

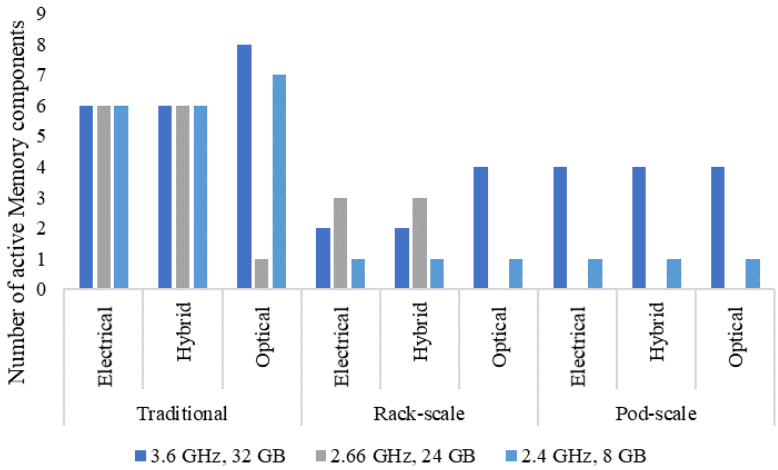


Figure 4.10: Active memory components under 20 CPU intensive workloads

For varying number of CPU intensive workloads, the TDPC in the traditional DC with electrical or hybrid network topology is largely subjective to TNPC. This is because the TCPC and TMPC are relatively constant. Hence, a traditional DC that deploys an electrical network topology has the highest TDPC. The TDPC of traditional DCs with hybrid network topology follows. The TDPC in the traditional DC with optical network topology is the lowest. This is because the adoption of optical switches in the topology ensures lower TNPC. This consequently encouraged lower TCPC and TMPC as observed when 20 CPU intensive workloads are provisioned. Under varying number of CPU intensive input workloads considered, 10% average percentage reduction in TDPC was obtained when the hybrid network topology replaced the electrical network topology. Similarly, 27% average percentage reduction in TDPC was obtained when the optical network topology replaced the hybrid network topology under varying number of CPU intensive input workloads considered.

Note that in the traditional DC, strict resource locality (required between CPU and memory components that host each workload) and resource capacity constraints collectively limits effective clustering of memory demands of workloads. Memory demands of workloads in the same workload group are often placed in different same memory component. Hence, memory data shuffle traffic between memory components in different nodes has little impact on the placement of workloads in traditional DC when energy minimisation is the goal. Memory data shuffle traffic exists between nodes if two workloads within the same workload group, with inter-memory traffic ($\mathcal{M}_{sd}$), are placed in different nodes.

Under varying number of CPU intensive workloads considered, physical disaggregation of traditional servers at rack-scale improves efficient usage of

memory resources relative to the traditional DC. Proportional usage of memory resources when disaggregation is implemented under all network topologies enabled this. Proportional usage of memory resources enables the reductions in the number and diversity of active memory resource components as shown in Figure 4.10. Consequently, corresponding reductions in TMPC is also achieved as shown in Figure 4.6, Figure 4.7 and Figure 4.8. Both number of active memory resource components and their corresponding TMPC reduced by more than 50% when traditional DC servers are physically disaggregated at rack-scale under all network topologies considered. Improvements in average active memory resource utilisation and energy efficient (i.e., proportional) usage of active memory resources in the DC is responsible for this. Consequently, 32 GB and 24 GB memory components are only activated when they can be highly utilised. Otherwise, 8 GB memory components are activated if memory capacity constraints permit. The relaxation of the inter-resource locality constraint in rack-scale disaggregated DCs enabled such flexibility.

The CPU intensive nature of input workloads implies that there are limited opportunities to improve overall CPU power efficiency in the DC via physical disaggregation. For example, the scenario where 20 CPU intensive workloads were optimally provisioned in the DC with electrical, hybrid or optical network topology. The results show that there is no improvement in TCPC of rack-scale DC relative to the value obtained when a traditional DC with a similar network topology was used. As reported under traditional DC, the type of network topology adopted also determines optimum placement of workload resource demands in rack-scale DCs after physical disaggregation. Hence, relative to the electrical or hybrid network topologies in a rack-scale DC, placement of workloads demands changes when the optical network is deployed. This revision leads to lower TCPC (6% fall) and (further decrease in) TMPC (5%) when the optical network topology is used with minimal increase in TNPC. However, this revised placement of CPU and memory resource demands which delivers better energy efficiency may increase inter-rack and inter-pod network traffic. Because of additional inter-memory data shuffle traffic between memory components of workloads that belong to the same workload group, which are placed in different racks (or pods). In summary, in rack-scale DC where resource locality is limited to the rack, deployment of optical network topology achieves lower TMPC. This is achieved at the cost of higher network traffic and marginal increase in TNPC. In contrast, when the hybrid or electrical network topology is deployed, lower network traffic and TNPC is preferred at the cost of higher TMPC.

In the rack-scale DC, high bandwidth traffic between CPU and memory components is rack-limited while low bandwidth DC north-south communication traverses higher tiers of the network topology adopted. Hence, as observed in the traditional DC, the electrical network topology has the highest TNPC in rack-scale DC. The TNPC of hybrid and optical network topologies follow in descending order as shown in Figure 4.6, Figure 4.7, and Figure 4.8. There is an increase in the number of instances in which memory demands of workloads (that belong to the same workload group) are collocated within the same memory component or the same homogenous memory node. This is because of the inter-resource locality constraint between CPU and memory components is relaxed and also because of the low memory demand of CPU intensive workloads. This consequently enables marginal reductions in TNPC of electrical, hybrid and optical network topologies alike under the rack-scale DC.

In the pod-scale DC, high-bandwidth traffic between CPU and memory components is pod-limited while low bandwidth DC north-south communication traverses all tier of the network topology in the DC. Hence, power consumption of electrical and hybrid network topologies in pod-scale DC increases significantly relative to traditional and rack-scale DCs as shown in Figure 4.6, Figure 4.7, and Figure 4.8. Sole adoption of power-hungry electrical switches in the electrical network topology resulted in very high TNPC relative to hybrid and optical network topologies as shown in Figure 4.6, Figure 4.7, and Figure 4.8. Unlike observation under traditional and rack-scale DCs, network topology does not inhibit optimal selection of CPU and memory resources in pod-scale DC. This is because all CPU-memory traffic must traverse inter-rack fabric due to the use of homogenous resourced racks in the pod-scale DC. As observed in the rack-scale DC, network power consumption resulting from inter-workload memory data shuffle is also significantly limited in pod-scale DC. This is achieved by placing memory demand of workloads of the same workload group in the same component or node. However, clustering of workloads memory demands into memory components or nodes also depends on memory capacity constraint, the power consumption of memory components and their corresponding impact on the total DC power consumption.

Comparison of TCPC and TMPC in rack-scale and pod-scale DCs shows that rack-scale DC can achieve similar performance as pod-scale DC as shown in Figure 4.6, Figure 4.7, and Figure 4.8. This is because the number and diversity of each resource type in each rack of rack-scale DC may be

sufficient (e.g., when 5 or 10 CPU intensive workloads are optimally provisioned) to enable optimal benefits of resource disaggregation. However, greater diversity and higher number of CPU resource components in homogenous racks of pod-scale DC relative to the situation in heterogeneous racks of rack-scale DC can enable better overall CPU power efficiency in pod-scale DC relative to the rack-scale DC. This is observed when 20 CPU intensive workloads are optimally provisioned in pod-scale DC with hybrid or electrical network topology. Hence, if the number and diversity of each resource type required by workloads is guaranteed during resource component allocation in rack-scale DC, similar TCPC and TMPC can be achieved under rack-scale and pod-scale DCs. In DCs that deploy electrical or hybrid network topology, satisfaction of the outlined requirement ensures lower TNPC in rack-scale DCs relative to the TNPC of pod-scale DCs. On the other hand, if an optical network topology such as EVROS is deployed, rack-scale DC's TCPC, TMPC and TNPC can match those of the pod-scale DC. This is observed when 10 CPU intensive workloads are provisioned in both rack-scale and pod-scale DCs that deploy the optical network topology and have equal number of active racks as seen in Figure 4.8.

Generally, under varying number of CPU intensive workloads, the highest (7%) average percentage reduction in TDPC is achieved when rack-scale DC with optical network topology is adopted to replace the traditional DC with optical network topology. Compared to the traditional DC with hybrid network topology, 3% average percentage reduction in TDPC is achieved when rack-scale DC with hybrid network topology is adopted to replace a traditional DC with the same network topology. Compared to the traditional DC with electrical network topology, the average percentage reduction in the TDPC of rack-scale DC with similar network topology is 2%. It is also observed that only the optical network topology enabled reductions in the TDPC of traditional DC when replaced with the pod-scale DC (except when 5 CPU intensive workloads are optimally provisioned). With respect to reduction in TDPC, the results also show that physical disaggregation should be limited to the rack–scale to ensure TDPC reductions if electrical or hybrid network topology are to be deployed for disaggregated DC. Going further by disaggregating at pod-scale leads to significant (up to 50%) increase in the TNPC of electrical and hybrid network topologies. This surpasses any savings in TCPC and TMPC derived from physical disaggregation of traditional DC at pod-scale.

4.4.2 Memory Intensive Workloads

Results under memory intensive workloads further highlights trends identified when CPU intensive workloads were provisioned. Proportional usage of CPU and memory resources in disaggregated DCs (as seen in Figure 4.11) led to lower TCPC and TMPC as seen in Figure 4.12, Figure 4.13 and Figure 4.14. For instance, when 5 input workloads are provisioned across traditional, rack-scale, and pod-scale DCs, physical disaggregation of resources at rack-scale and pod-scale leads to reductions in TCPC and TMPC relative to the traditional DC. This is achieved at the expense of higher TNPC for all network topologies considered as observed when CPU intensive workloads were considered.

Relative to similar results obtained when CPU intensive workloads were considered, Figure 4.11 shows that memory components are highly utilised under all DCs. This is because of the memory intensive nature of the input workloads. The average utilisation of CPU resource components is lower (about 60%) in traditional DCs compared to about 97% average utilisation obtained when physical disaggregation of resources at rack-scale and pod-scale are employed. The average active utilisation of CPU and memory components are 60% and 87% respectively when memory intensive workloads are deployed in the same DC as shown in Figure 4.11. On the other hand, the average utilisation of CPU and memory components are 84% and 48% respectively when CPU intensive workloads are deployed in traditional DCs as shown in Figure 4.5. Similarly, the average active utilisation of CPU and memory resource components are 97% and 87% respectively when memory intensive workloads are deployed in physically disaggregated DCs as seen in Figure 4.11. Relatively, as shown in Figure 4.5, 86% and 93% are the average active resource utilisation of CPU and memory components respectively when CPU intensive workloads are deployed in physically disaggregated DCs. Hence, greater proportional usage of DC resource components is achieved when memory intensive workloads are provisioned in both traditional and physically disaggregated DCs compared to when CPU intensive workloads are provisioned.

The power consumption profiles of switching components also affects the placement of workload’s memory resource demands in rack-scale DC. This is consistent with the observation made when CPU intensive workloads were provisioned. This is because the trade-off between TMPC and the volume of inter-memory data shuffle traffic remains an important criterion for optimal placement of memory resource demands in rack-scale DCs. It can be

observed that lower TMPC is achieved when memory intensive workloads are provisioned in a rack-scale DC with optical network topology. However, it is achieved at the expense of higher network (inter-memory data shuffle) traffic and marginal increase in TNPC. In contrast, a rack-scale DC with hybrid or electrical network topology reduces network traffic to achieve lower TNPC by reducing inter-memory traffic in the network as the expense of higher TMPC.

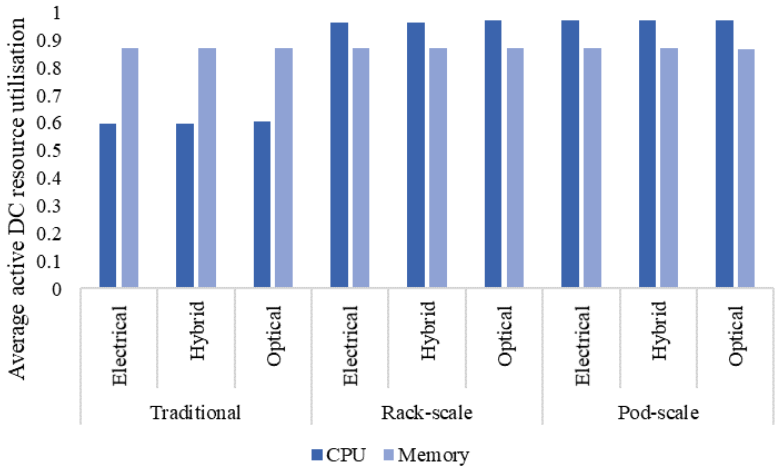


Figure 4.11: Average utilisation of active DC resource components under 20 memory intensive workloads

Comparison of TCPC and TMPC obtained in rack-scale and pod-scale DCs under memory intensive workloads also shows that physical disaggregation at rack-scale can achieve equal performance in terms of CPU and memory power efficiencies as physical disaggregation at pod-scale. This is possible (as explained previously) when resource allocation ensures that both CPU and memory resources are available in the appropriate diversity and number in each rack in the rack-scale DC. This requirement is satisfied when 5 memory intensive workloads are provisioned in rack-scale and pod-scale DCs as shown in Figure 4.12, Figure 4.13 and Figure 4.14. These input workloads require one 3.2 GHz and two 2.4 GHz CPU components and one 32 GB and two 24 GB memory components for optimal placement. Hence, they can be adequately provisioned within a single rack of the rack-scale DC to achieve the same optimal TCPC and TMPC observed in the pod-scale DC. Results obtained under 20 memory intensive workloads also reveal the strong impact of capacity constraint during workload placement as shown in Figure 4.15 and Figure 4.16. For instance, because the memory resource demand of all 20 memory intensive workloads exceed 8 GB, the 8 GB memory components do not have sufficient capacity. Therefore, there is no active 8 GB memory component in the DC as seen in Figure 4.16.

As expected, TNPC of pod-scale DC with electrical/hybrid network topology increases significantly relative to the TNPC of rack-scale DC with electrical/hybrid network topology. Note that the TNPC of the electrical network topology is always higher than that of the hybrid network topology. On the other hand, if optical network topology is adopted in both rack-scale and pod-scale DCs and the number of active racks in both DCs are equal, the TNPC of the rack-scale DC is approximately equal to the TNPC of the pod-scale DC (e.g. under 10, 15 and 20 memory intensive workloads). If the number of active racks in rack-scale DC is less than the same number in pod-scale DC, then the additional optical switch required per additional active rack is responsible for most of the difference in TNPC as shown in Figure 4.14. This is the case when 5 memory intensive workloads are provisioned.

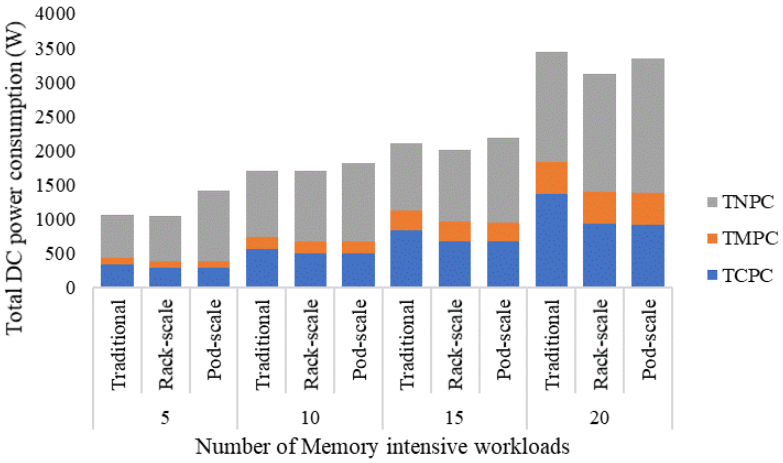


Figure 4.12: Total DC power consumption under memory intensive workloads when electrical network topology is deployed

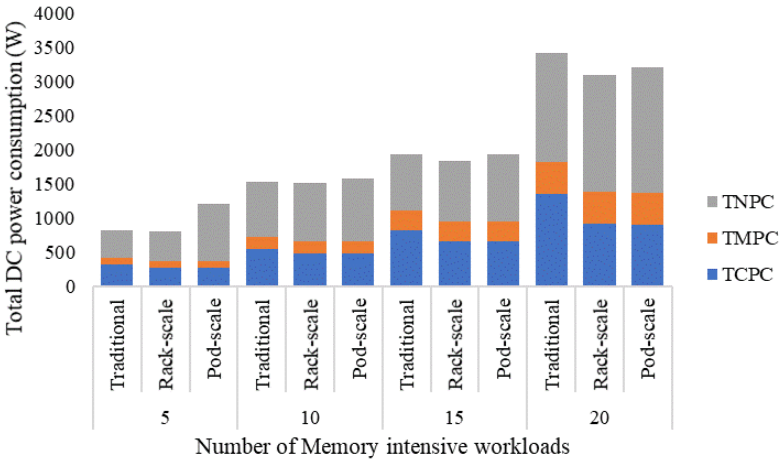


Figure 4.13: Total DC power consumption under memory intensive workloads when hybrid network topology is deployed

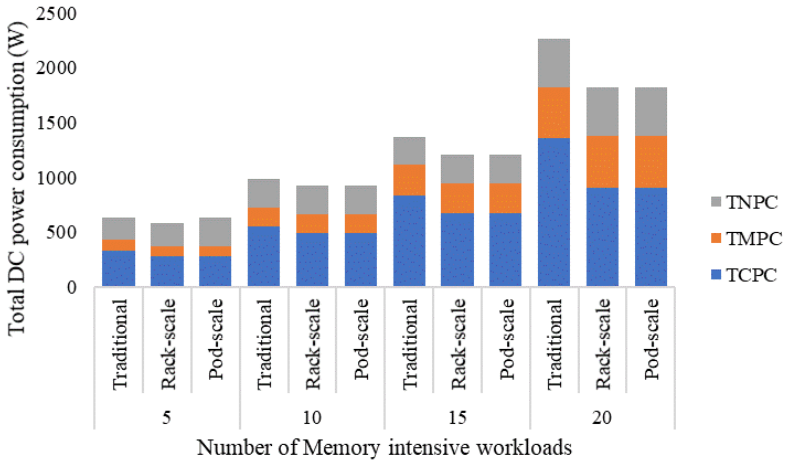


Figure 4.14: Total DC power consumption under memory intensive workloads when optical network topology is deployed

Compared to the traditional DC with optical network topology, rack-scale DC with optical network topology reduces the TDPC by up to 6-20% under varying number of memory intensive workloads as shown in Figure 4.14. This is in contrast with 5-8% reduction in TDPC observed when CPU intensive workloads were provisioned in a similar setup. Lower CPU resource demand of memory intensive workloads is responsible for the savings TDPC observed. Relative to the traditional DC with optical network topology, the TDPC of pod-scale DC with optical network topology is also often lower under varying number of memory intensive workloads as shown in Figure 4.14. For example, when 5 memory intensive workloads are optimally provisioned the TDPC in the pod-scale is less than that of traditional DC. In this case, only one rack is active in traditional DC while 2 active racks are required in pod-scale DC.

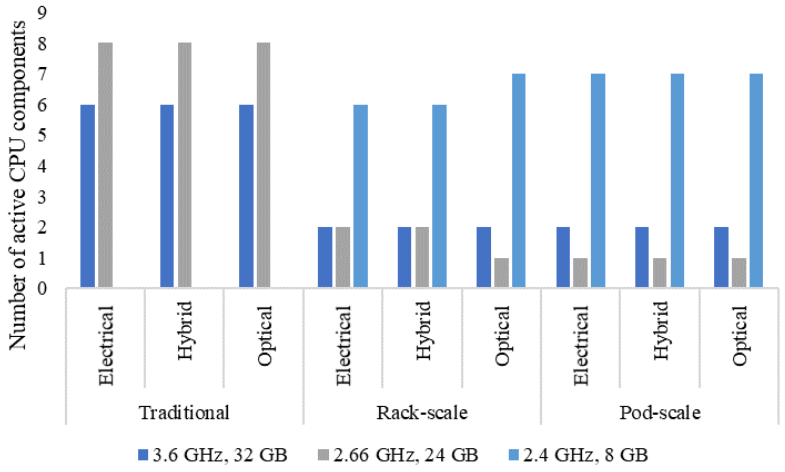


Figure 4.15: Active CPU components under 20 memory intensive workloads

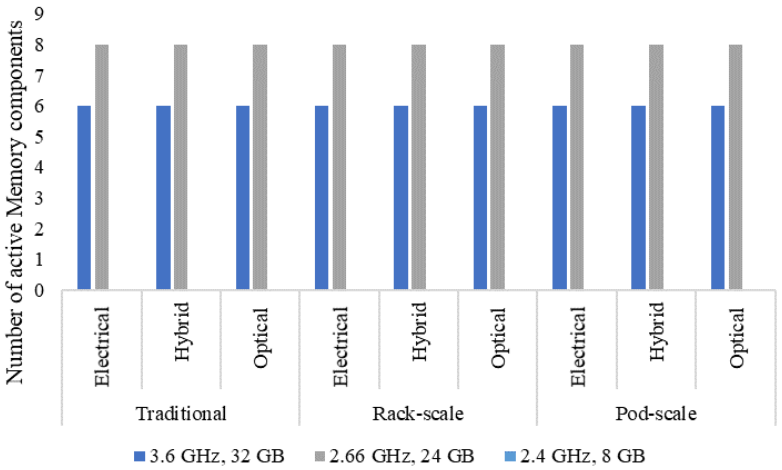


Figure 4.16: Active memory components under 20 memory intensive workloads

Relative to the TDPC of traditional DC that deploy electrical or hybrid network topology, the TDPC of rack-scale DC with similar network topology is also lower as shown in Figure 4.12 and Figure 4.13. Likewise, relative to the TDPC of traditional DC that employ hybrid network topology, the TDPC of pod-scale with similar network topology is occasionally (e.g. when 15 and 20 memory intensive workloads are provisioned) lower as shown in Figure 4.13 (except when 5 memory intensive workloads are optimally provisioned). Figure 4.12 also shows that the TDPC of pod-scale DC with electrical network topology is often (e.g. under 5, 10 and 15 memory intensive workloads) higher than that of traditional DC with a similar network topology. This is because of significant increase in TNPC which exceeds savings in TCPC and TMPC. However, as the number of workloads increases (i.e. 20 memory intensive workloads), the idle power of electrical switches in network topology is increasingly shared. Hence. their corresponding impact is reduced.

Comparison of TDPC in Figure 4.6 - Figure 4.8 to the TDPC in Figure 4.12 - Figure 4.14 shows that the TDPC is lower in most scenarios when memory intensive resources are provisioned in corresponding DC type and network. As expected, the TCPC decreases when memory intensive workloads are deployed instead of CPU intensive workloads in all DC and network types. On the other hand, TMPC increases accordingly in all DC and network types. The TNPC is often higher when memory intensive workloads are deployed instead of CPU intensive workloads in corresponding DC and network types. Since the same inter-memory shuffle traffic and inter-resource traffic are adopted when CPU or memory intensive workloads are deployed, high intensity of memory resource demand of input workloads is responsible for this trend. This is because opportunities to consolidate the memory

resource demands of workloads that are in the same group into the same memory component or node are reduced when memory intensive workloads are deployed. Hence, more traffic traverses higher network tiers compared to the situation when CPU intensive workloads with finer memory resource demands are deployed in traditional and rack-scale DCs. Relative to traditional and rack-scale DCs, increase in network traffic is minimised in pod-scale DCs. This is because the adoption of homogenously resourced racks reduces the increase of network traffic which could be attributed to higher memory resource intensity. Furthermore, the impact of increased network traffic on the TNPC is low when the optical network topology is deployed in DCs compared to when electrical or hybrid topologies are deployed. This is because of electrical switches in both electrical and hybrid network topologies. These electrical switches have a power consumption profile that is proportional to the volume of traffic in the network.

In spite of the marginal benefits enabled by pod-scale over rack-scale resource disaggregation, empirical work by authors in [106] shows that average performance degradation increases when CPU and memory resources are physically disaggregated beyond rack-scale. Moreover, an optical-circuit-switched (OCS) based network topology can also guarantee minimal access latency between disaggregated CPU and memory resource via temporal path reservation. Hence, the evaluations in subsequent section of this chapter adopts the rack-scale physical disaggregation as the maximum scale of resource disaggregation. The optical network topology is also adopted as the default network topology is succeeding sections of this chapter. This setup ensures minimal applications' performance degradation in DCs as the concept of logical disaggregation is explored in traditional DCs.

4.4.3 Logical Resource Disaggregation at Rack-Scale

As shown earlier, physical disaggregation of compute resources at rack-scale is sufficient to enable the required flexibility if DC resource allocation ensures that resources are available in appropriate number and/or diversity. This flexibility brings about improvements in resource utilisation and overall DC energy efficiency. It was also observed that the use of an optical network topology within a rack-scale DC ensures maximal network energy efficiency. Furthermore, optical networks are also expected to enable minimal increase in resource access latency between separated compute resources. On the other hand, optical network topologies are under-utilised in traditional DCs where high-bandwidth traffic is node-limited. This negatively impacts the energy efficiency of the optical network topology in a traditional DC. However,

features, such as high bandwidth and ultra-low latency communication, supported by the optical network topology can be effectively used in traditional DCs by relaxing the locality constraint to permit resource sharing within the rack. This enables a second approach to achieve rack-scale DC via logical resource disaggregation.

In a traditional DC, logical disaggregation implies that CPU and memory components remain in heterogeneous resourced nodes. However, individual access of each component over a suitable network is allowed and resource locality is confined to each rack in the DC to ensure latency related SLAs are enforced. This contrasts with a traditional DC where resource locality is confined to each node in the DC. Application of logical disaggregation to the traditional DC can enable performance that approaches those reported under the physically disaggregated rack-scale DC. The performance of logical disaggregation in traditional DC is compared to that of a physically disaggregated rack-scale DC using power consumption as the reference metric. The optical network topology is adopted under both scenarios.

As expected, the performance of the logically disaggregated traditional DC with heterogeneous nodes approaches that of a physically disaggregated rack-scale DC where homogeneous nodes are placed in heterogeneous racks. The results in Figure 4.17 and Figure 4.18 show that the TDPC of a logically disaggregated traditional DC is marginally lower than that of a physically disaggregated rack-scale DC for varying number of CPU and memory intensive workloads. The marginal fall in TNPC due to better network utilisation efficiency in logically disaggregated traditional DC is responsible for the marginal decrease in TDPC. This is because CPU and memory components are co-located within the same node in the logically disaggregated traditional DC. Hence, remote memory access over intra-rack network fabric is occasionally prevented. Thus, enabling better network energy efficiency that lead to lower TNPC. This is not possible in physically disaggregated rack-scale DC. Relative to the physically disaggregated rack-scale DC, the logically disaggregated traditional DC can also guarantee SLAs for ultra-latency sensitive workloads. This is because logically disaggregated traditional DC can default to a traditional DC infrastructure in extreme scenarios to satisfy SLAs while a physically disaggregated rack-scale DC cannot. Note that an extreme scenario implies that workloads with very high-sensitivity to increase in memory access latency are being provisioned.

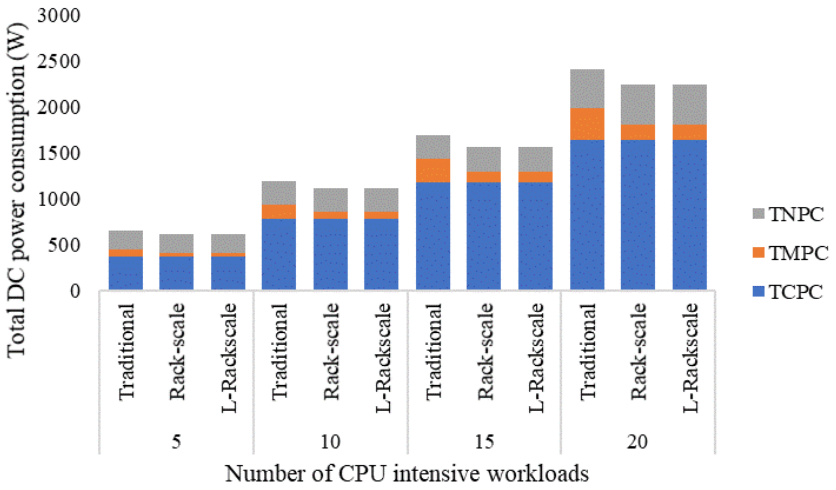


Figure 4.17: Total power consumption of physically and logically disaggregated rack-scale DCs under CPU intensive workloads

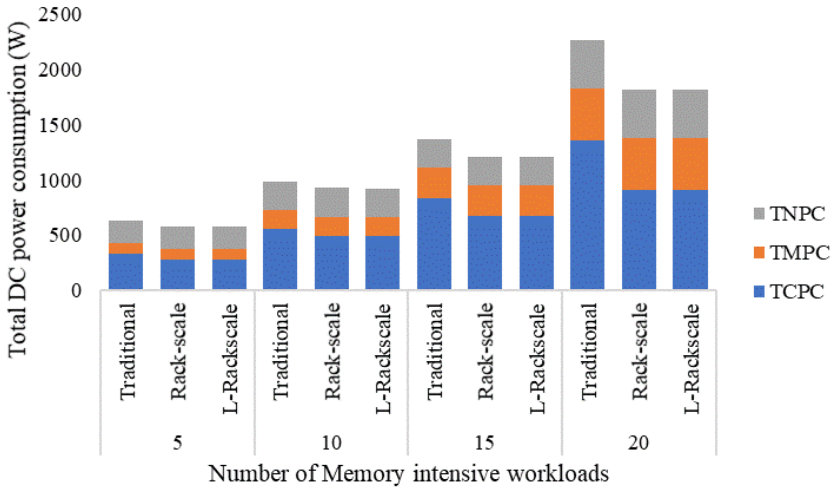


Figure 4.18: Total power consumption of physically and logically disaggregated rack-scale DCs under memory intensive workloads

The placement of CPU and/or memory resource demands in logically disaggregated traditional DC may differ from the placement in physical disaggregated rack-scale DC. Such placement disparity many necessary to ensure that the CPU-memory pairs used to provision some workloads are in the same node. Hence, the TCPC and/or TMPC under logically disaggregated traditional DC may marginally increase or decrease relative to the TCPC and/or TMPC under physically disaggregated rack-scale DC. Additionally, if logical disaggregation of traditional DC server is to be adopted, modular resource component design must be adopted in such servers. This will ensure better resource upgrade lifecycle relative to conventional traditional server architecture.

4.5 Micro-Service Architecture in Composable DCs

In the previous sections, monolithic workloads were considered as input workloads for the variety of composable DCs studied. A monolithic workload has fixed resource demands and is designed to run on bespoke physical or virtual hardware for peak performance as seen in previous sections. As an alternative workload architecture, the micro-service architecture, proposes the decomposition of monolithic workloads into independent components called micro-services. A micro-service perform a specific business function and can be developed, tested, deployed, managed and scaled individually [38]. Related micro-services performing different business functions are thereafter provisioned concurrently to form an integrated workload which is comparable to the decomposed monolithic workload. For example, a monolithic e-commerce workload which comprise of different intrinsic units performing accounting, inventory and ordering functions can be decoupled into 3 independent micro-services. The 3 micro-services form a single integrated workload.

Communication between micro-services associated with the same integrated workload is facilitated using well-defined standards or application programming interfaces (APIs) [39]–[41]. The APIs isolates inner workings of each micro-service. This novel approach can further enhance scalability, agility, and resource utilisation in DCs. In this section, the use of micro-services to create integrated workloads in composable DCs (with pre-allocated heterogeneous CPU and memory resources) is compared to the use of monolithic workloads.

4.5.1 MILP Model Extension

The model in Section 4.3.2 is extended by introducing set, parameters, variables, and constraints to establish the relationship between an integrated workload and its micro-services. The additional model set, parameters and variable are given as follows.

Sets:

I	Set of integrated workloads

Parameters:

CI_i	CPU capacity of integrated workload i
MI_i	Memory capacity of integrated workload i

WI_{wi} Indicates the relationship between a micro-service workload w and integrated workload i. $WI_{wi} = 1$ if micro-service workload w is associated with integrated workload i. Otherwise, $WI_{wi} = 0$

Variables:

$\mathbb{i}_{\boldsymbol{i}}$ Indicates the state of integrated workload i i.e., served or unserved. $\mathbb{i}_{\boldsymbol{i}} = 1$ indicates the integrated workload i is served. Otherwise, $\mathbb{i}_{\boldsymbol{i}} = 0$

The model in Section 4.3.2 (where set W represents a set of monolithic workloads) continues to represent the DCs supporting monolithic workloads. On the other hand, in DCs supporting micro-services, integrated workloads are represented by set I while set W is the set of micro-services. A group of micro-services make an integrated workload. Based on the results from previous sections of this chapter, impact of micro-services is evaluated in a logically disaggregated traditional DC. Hence, representing a composable DC infrastructure that can support the SLA requirements for both latency sensitive and non-sensitive workloads. The OCS-based optical network topology is adopted. Hence, inter-memory and inter-resource communication has little impact on workload resource placement as reported in the previous sections. The MILP model is further simplified by excluding variables and parameters required to estimate inter-memory traffic flows. This is because the results from the earlier sections show that inter-memory traffic has limited impact on workload placement over OCS-based optical network topology.

In addition to constraints (4.14) – (4.31), the following constraints establish the relationship between integrated workloads and micro-services in a composable DC.

$$\sum_{w \in W} \sum_{j \in C} WC_w \, \boldsymbol{c_{wj}} \; WI_{wi} = CI_i \, \mathbb{i}_{\boldsymbol{i}} \qquad (4.36)$$

$$\forall \, i \in I$$

$$\sum_{w \in W} \sum_{j \in M} WM_w \, \boldsymbol{m_{wj}} \, WI_{wi} = MI_i \, \mathbb{i}_{\boldsymbol{i}} \qquad (4.37)$$

$$\forall \, i \in I$$

Constraints (4.36) and (4.37) ensure that an integrated workload is served only if all micro-services associated with the integrated workloads are served. Otherwise, the integrated workload resource demand is rejected. These constraints apply only when micro-services are being provisioned.

4.5.2 Evaluation Scenarios and Results

The resource allocation and resource classes described for the traditional DC from earlier sections are maintained. Similarly, the 20 CPU and memory intensive monolithic workloads used in earlier sections are adopted to represent integrated workloads. It is assumed that each integrated workload (CPU or memory intensive) is a unit comprising of two related micro-services that can function independently. Hence, to study the impact of increased workload modularity in composable DCs, each integrated workload is decoupled into two independent micro-service workloads. It is conservatively assumed that the in/out-bound inter-resource communication traffic for each integrated workload is equally shared between its intrinsic micro-services. Table 4.5 gives an illustration of all possible evaluation setups for both CPU and memory intensive input workload classes, the acronyms outlined in the Table 4.5 are used to represent each setup hereafter.

Table 4.5: List of evaluation setups

Setup	**Resource allocation**	**Resource disaggregation**	**Workload architecture**
TS-Mono	Traditional server	**-**	Monolithic
RS-Mono	Traditional server	Logical	Monolithic
TS-Micro	Traditional server	-	Micro-service
RS-Micro	Traditional server	Logical	Micro-service

The impact of different workload architectures and resource disaggregation is studied in DCs. Evaluation metrics include DC resource power consumption, number of active DC resources and average active resource utilisation as illustrated in Figure 4.19, Figure 4.20, Figure 4.21 and Figure 4.22. The model achieves optimal results via bin-packing of workloads demands onto DC resources. The model attempts to achieve optimal resource power efficiency and utilisation efficiency within capacity and resource locality constraints.

Note that the placement of workload demands, TCPC and TMPC obtained under the both TS-Mono and RS-Mono setups when CPU or memory intensive input workloads are provisioned is like results obtained in earlier

sections where similar scenarios have been considered. There are marginal drops in TMPC and TNPC obtained under both TS-Mono and RS-Mono setups relative to values obtained previously. The absence of inter-workload traffic in the revised model is responsible for this observation. Thus, the discussion of results in this section is focused on the performance of TS-Micro and RS-Micro setups and the comparison of such performance to the results under TS-Mono and RS-Mono setups. To avoid repetition, the results of the TS-Mono and RS-Mono setups are only illustrated in Figures.

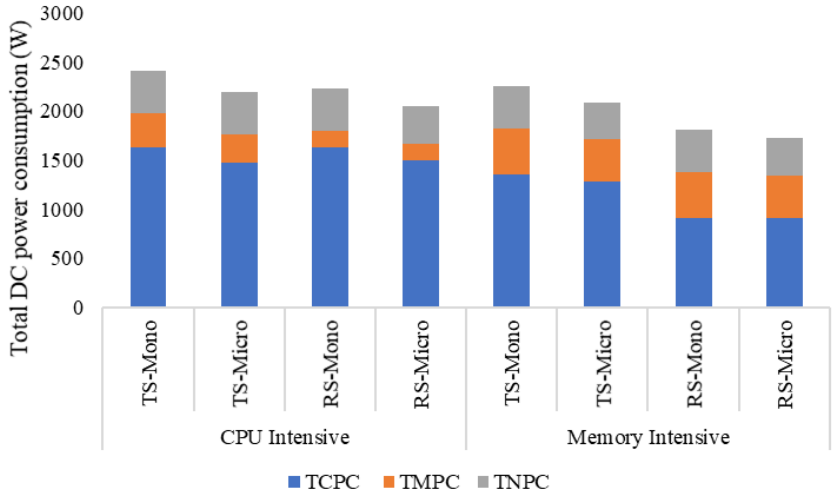


Figure 4.19: Power consumption of DCs

An inspection of the TDPC under the CPU intensive workload in Figure 4.19 shows that TS-Micro setup reduces the total resource power consumption by 9% compared to the TS-Mono setup. This is because of CPU intensive nature of input workloads and the dominance of CPU resource power consumption over other DC resource types. Reductions in CPU resource power consumption is responsible for more than half (74%) of savings made in total DC power consumption. Savings in total memory resource power consumption is responsible for further savings made. On the other hand, TNPC remains constant under both setups. A transit from TS-Mono setup to TS-Micro setup under CPU intensive workload leads to 10% and 16% reductions in power consumption of CPUs and memory components respectively. These reductions are enabled via increased workload modularity. Increase workload modularity leads to improved bin-packing of finer workload resource demands into a different configuration of active servers to achieve greater power efficiency as seen in Figure 4.21 and Figure 4.22.

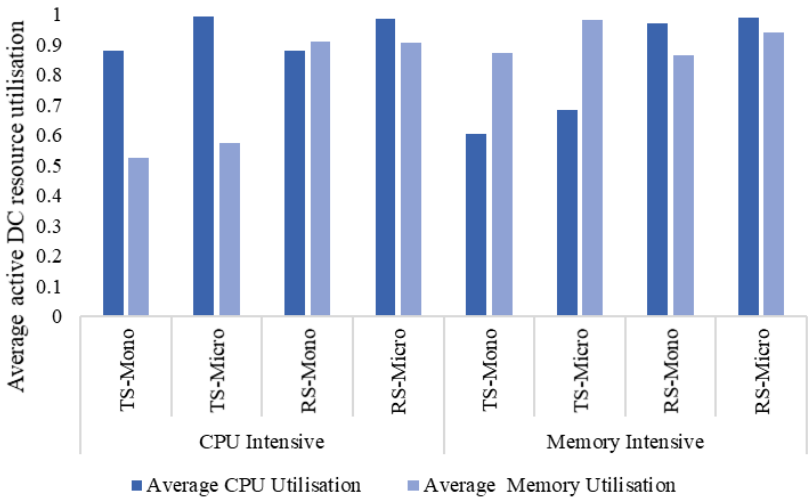


Figure 4.20: Average utilisation of active DC server resources

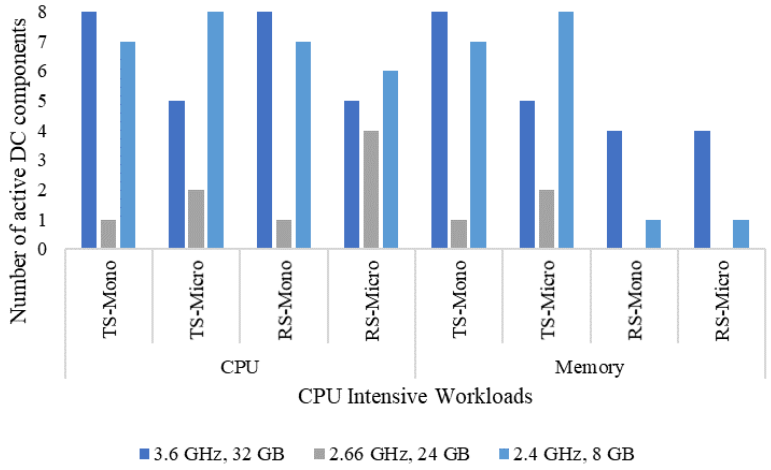


Figure 4.21: Active DC resources under CPU intensive workloads

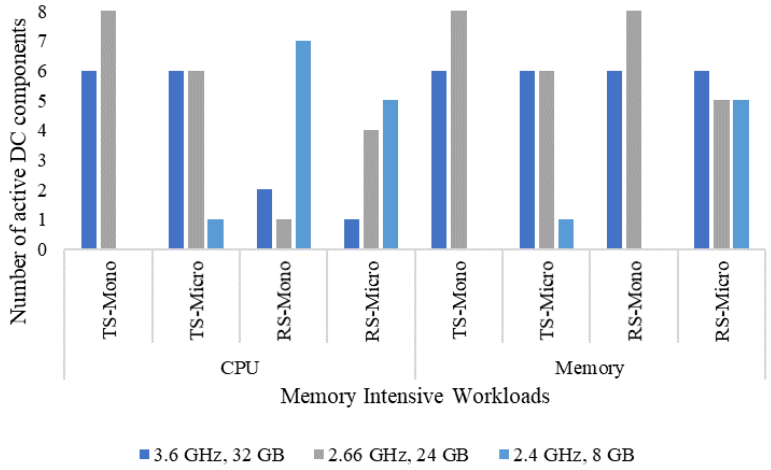


Figure 4.22: Active DC resources under memory intensive workloads

Under the TS-Mono setup, capacity constraint enforced the use of high-performance server to provision workloads with CPU resource demand that is

above 2.66 GHz. However, the finer resource demands under the TS-Micro setup enable improved bin-packing of demands into servers. This consequently improved active resource utilisation relative to the TS-Mono setup as shown in Figure 4.20. Nevertheless, it is important to note that CPU intensive nature of input workloads and the strict resource locality constraint of traditional servers led to the high number of active servers. Such servers have high average CPU utilisation and under-utilised memory resources. Therefore, disproportionate utilisation of DC resources under traditional DC architecture may persist even with increased workload modularity.

A transition to the RS-Mono setup for CPU intensive workloads results in 7% decrease in TDPC compared to the TS-Mono setup. The relaxation of inter-resource locality constraint in logically disaggregated servers is responsible for this observation. Reductions in the power consumption of memory components is solely responsible for the drop observed. The impact of this drop on the TDPC is slightly restricted by the marginal rise in the TNPC (as shown in Figure 4.23). The TNPC increased because of the increase in the volume of traffic traversing on-board and intra-rack networks.

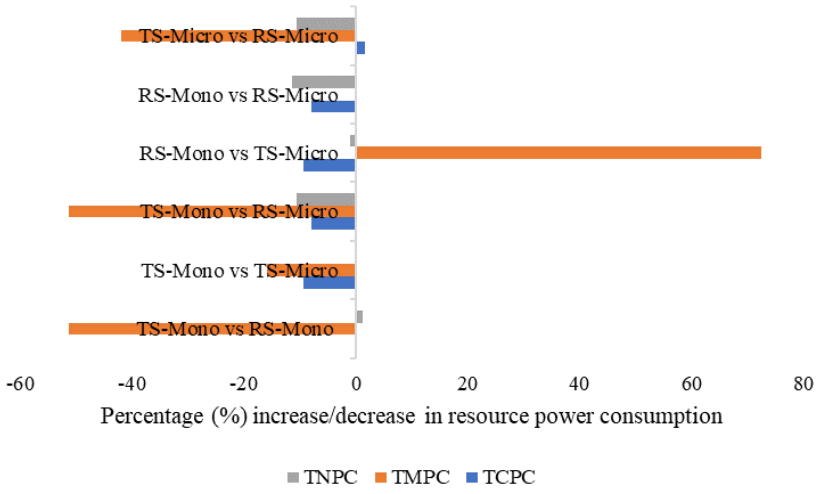


Figure 4.23: Percentage increase or decrease in DC resource power consumption of CPU intensive workloads

Relative to the TS-Mono setup, logical disaggregation in the RS-Mono setup enabled 51% reduction in TMPC at the expense of 1% rise in TNPC as shown in Figure 4.23. When monolithic workloads are considered under the RS-Mono setup, the same TCPC reported under the TS-Mono setup is obtained despite logical server-disaggregation. This is because improved packing of workloads' intensive CPU demands was not feasible. This is sub-optimal, relative to 10% reductions in TCPC achieved under TS-Micro setup as shown in Figure 4.23. Significant reductions in the TMPC under the RS-

Mono setup relative to the TS-Mono setup is achieved via consolidation of granular memory resource demands onto reduced number of highly utilised memory components as shown in Figure 4.20 and Figure 4.22. CPU components are also highly utilised as shown in Figure 4.20. However, this is because of the CPU intensive nature of workload demands and not because of improved consolidation. Notwithstanding, disaggregation addresses disproportionate usage of DC resources seen in traditional DC infrastructure. This is because greater resource modularity enabled by disaggregation increases flexibility when CPU and memory components used to provision workload demands are selected. Hence, only the minimum number of resources needed to effectively satisfy each type of workload resource demand are activated.

Adoption of both logical disaggregation and micro-service architecture under the RS-Micro setup addresses the limitation of RS-Mono setup to achieve additional savings in TDPC. Relative to TS-Mono, TS-Micro and RS-Mono setups, RS-Micro leads to 15%, 7% and 8% reduction in total DC power consumption respectively. Thus, the RS-Micro setup activates unattained potentials of TS-Mono, TS-Micro and RS-Mono setups for optimal DC efficiency via improved bin-packing of fine resource demands onto active resource components. For example, relative to the RS-Mono setup, a move to the RS-micro leads to 8% reduction in TCPC. This power savings is achieved via the use of a different configuration of active CPU components as shown in Figure 4.21. At the same time, only marginal reduction in TMPC is achieved when RS-Micro setup is adopted over the RS-Mono setup. This is because the granularity of non-intensive memory resource demands is sufficient to achieve optimal memory power consumption in a disaggregated DC without adoption of the micro-service architecture.

Similar observations to those reported above for CPU intensive workload class are repeated under memory intensive workload classes. Hence, 7%, 20% and 23% reduction in the TDPC of the TS-Mono setup was achieved via the deployment of TS-Micro, RS-Mono and RS-Micro setups respectively for the memory intensive workload class.

The TS-Micro setup enables reductions in TCPC, TMPC and TNPC compared to the TS-Mono setup for memory intensive workloads as shown in Figure 4.24. When workloads are memory intensive, the TS-Micro setup is more power efficient for provisioning memory demands than the RS-Mono setup. On the other hand, the RS-Mono setup is more power efficient for provisioning CPU demand of memory intensive workloads compared to the

TS-Micro setup as shown in Figure 4.24. This is because the memory intensive nature of monolithic workloads limits optimal consolidation of such workload memory resource demand in the RS-Mono setup. Thus, resulting into higher memory power consumption under RS-Mono setup relative to the TS-Micro setup as shown in Figure 4.19 and Figure 4.24. However, reductions in the TCPC which significantly dominates memory resource utilisation inefficiencies leads to lower TDPC in RS-Mono setup relative to the TS-Micro setup when memory intensive monolithic workloads are provisioned. This is because of the dominance of CPU resource power consumption over memory resources. The RS-Micro setup addresses the limitation of the RS-Mono setup. Relative to the TS-Micro, RS-Micro can deliver equal or better TCPC and TMPC. Hence, surpassing the limitations of the RS-Mono setup as shown in Figure 4.19.

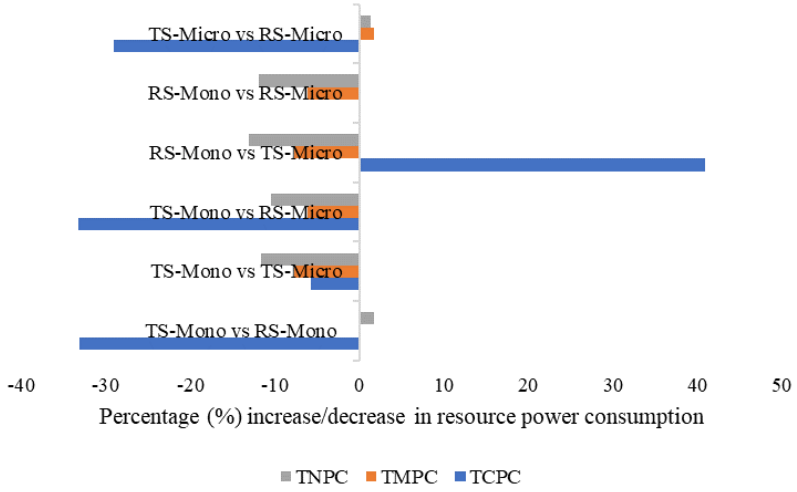


Figure 4.24: Percentage increase or decrease in DC resource power consumption of memory intensive workloads

When the class of input workloads is neither CPU intensive nor memory intensive, we expected limited reduction in the TDPC of the TS-Mono setup relative to TS-Micro, RS-Mono or RS-Micro setup. This is because the impact of capacity constraint is relaxed when workloads are not CPU intensive nor memory intensive. Likewise, lower resource intensity of such workloads implies that they are likely to approach optimal performance (i.e. energy efficiency) under the TS-Mono setup. Hence, only marginal performance gains will be achieved if such workloads are provisioned under TS-Micro, RS-Mono or RS-Micro setup.

4.6 Heuristic for Energy Efficient Placement of Workloads in Composable DCs

Results from the MILP model showed that efficient placement of monolithic and micro-service workloads in composable DCs is required to reach optimal energy efficiency in DCs with heterogeneous resource types. This task which was formulated as MILP model is a multi-dimensional bin packing problem with multi-sized bins of different resource types which may or may not be co-located. The classical one-dimensional bin-packing problem is NP-complete, hence multi-dimensional bin packing problem with bins which have varying capacity and power consumption is also NP-complete. Therefore, only approximation algorithms that mimic the results of the MILP model for different workload classes can provide best-fit solutions without the need for exhaustive search and permutation of workloads and DC resources. To this end, a heuristic for energy efficient placement (HEEP) of workloads in a rack-scale composable DCs is proposed.

4.6.1 HEEP Algorithm Description

HEEP is a greedy algorithm designed to be deployed with a centralised or distributed DC orchestration and control platform. It is required that the orchestration platform maintain global knowledge of resource state, utilisation, and power consumption across the DC as in software defined DCs. The orchestrator should also be aware of input workload resource demands over a specific time frame. The orchestrator is responsible for energy efficient placement of different workload classes in different DC architectures. The flow chart of the HEEP algorithm is given in Figure 4.25.

Given a set of input workloads with CPU or memory resources demands with inter-resource communication traffic, the algorithm achieves minimal total DC resource power consumption through the following steps. Input workloads are arranged in descending order of CPU/memory resource demands with respect to the workload class under consideration. This minimises blocking of workloads with ultra-high resource demands since the heuristic adopts a greedy approach. The sorted list of workloads is the input to the algorithm and is called the job list hereafter. The highest ordered workload in job list is set as the default query workload. The central orchestrator adopts a divide and conquer approach. It searches all nodes across the DC for the best CPU and memory resource component within each node to provision the query workload. If feasible, each node returns a candidate CPU and/or memory component to provision the query workload

alongside the corresponding power consumption and utilisation of such component in the candidate node.

Note that if a rack with sufficient CPU and memory resource capacities to host the query workload does not exist (i.e., the workload blocking criterion) the query workload is blocked and removed from the job list. Otherwise, the central orchestrator selects the best CPU and memory components to provision the query workload in each rack. Selections are made using pre-defined utilisation thresholds for CPU and memory components as illustrated in Figure 4.26. Afterwards, the central orchestrator also selects the best rack to provision the query workload in each pod based on the utilisation threshold of the best candidate CPU component of each rack. Finally, the central orchestrator selects the best pod to provision the query workload also based on the utilisation threshold of the best candidate CPU component in the best candidate rack of each pod. Afterwards, the query workload is removed from the job list. These steps imply that the algorithm searches all nodes, racks, and pods to obtain the best candidate resource components to host each workload resource demand.

If the job list is not empty, the heuristic attempts to select the next query workload by scanning through the job list for a workload that will fit into unused capacities of active resources in the present best rack to increase utilisation. For resource intensive workloads, the scan attempts to fill the idle CPU capacity of the present best CPU component. This is because CPU components have higher peak power consumption compared to memory components. If the input workload class is CPU or memory intensive, the scan orders workloads in descending order of CPU resource demand intensity. Thus giving higher preference to CPU resource demand. Higher preference is also given to CPU resource demands when memory intensive workloads are being considered. This ensures CPU biased resource allocation which leads to optimal total DC resource power consumption.

A successful scan returns a workload which becomes the new query workload that is placed in the present best rack. Otherwise, the highest ordered unserved workload in the job list is selected as the new query workload. When all workloads have been successfully placed or blocked, the algorithm estimates the total network power consumption resulting from workloads' CPU and memory resource demand placement across the DC, and subsequently reports the total DC resource power consumption before stopping. Note that in the algorithm, ties are always broken by selecting the first item (i.e. workload, resource component or rack) that appears.

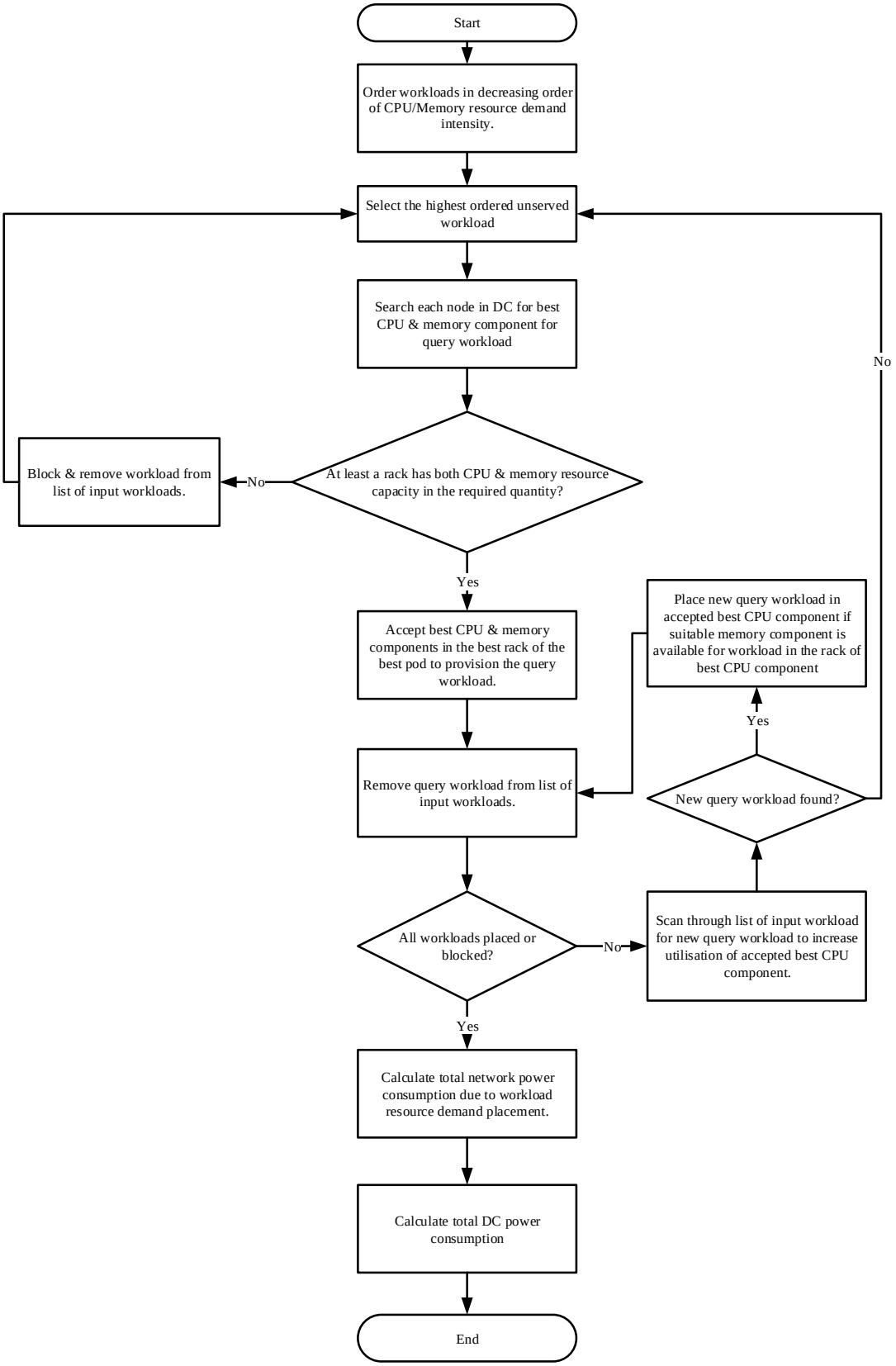


Figure 4.25: HEEP algorithm flow chart

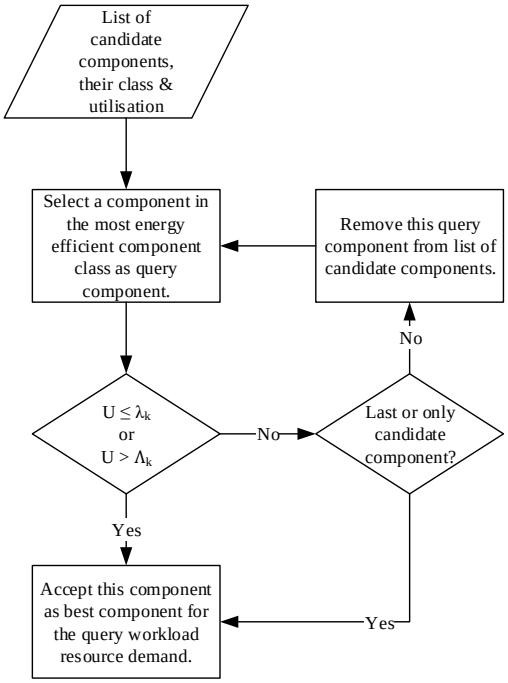


Figure 4.26: Best component selection based on component's class utilisation

Given a list of candidate components for a specific resource demand type, their component class and the resulting utilisation of placing the resource demand in the candidate components, Figure 4.26 gives an illustration of the steps taken to select the best components. Selections are made based on relative utilisation thresholds of each component class. A component in the most energy efficient component class (in the list of candidate components) is given priority. However, such component is only adopted as the best component under the following conditions:

- If the corresponding utilisation (U) after the placement of the query resource demand is less than or equal to the lower utilisation threshold (λ_k) defined for that component's class k.
- If the corresponding utilisation (U) after the placement of the query resource demand is greater than the upper utilisation threshold (Λ_k) defined for that component's class k.
- If the component is the last (and/or only) component in the list of candidate component being evaluated to support the query resource demand.

Otherwise, the most energy efficient component is removed from the list of candidate components for the query resource demand. Subsequently, a new component is selected from the class of the most energy efficient component in list of candidate components. The lower and upper utilisation threshold must

be defined based on the relative energy efficiency of classes of resource components deployed in the composable DC.

Using the CPU or memory resource component classes listed in Table 4.1, the lower and upper utilisation thresholds for resource component classes are defined as follows.

- Given a set of candidate component classes K for a given resource type which is sorted in descending order of energy efficiency.
- The lower utilisation threshold (λ_k) of all component classes (except the last component class) in the set is always 0.5. This is because utilising 50% (or less) of a component class's capacity to provision the most intensive resource demand leaves half (or more) of the class's capacity in an idle state. This idle capacity can be utilised to provision another intensive resource demand.
- The upper utilisation threshold (Λ_k) of all component classes (except last component) in the set is defined by Equation (4.38). This is because to energy efficiently utilise a resource component of class k which has higher capacity and power consumption than another resource component of class $k+1$, the resource demanded by all workloads placed in the component with greater power efficiency must be greater than the capacity of the component with lower power efficiency.

Consequently, if it is assumed that all CPU or memory resource component classes listed in Table 4.1 are present in a list of candidate components for a given query workload, the relative utilisation thresholds of all candidate component classes (except last component) are as given in Table 4.6.

$$\Lambda_k = \frac{Capacity_{k+1}}{Capacity_k} \qquad (4.38)$$

$$\forall k \in K : k \neq |K|$$

Table 4.6: Relative utilisation threshold for component classes

$k \in K$	CPU Capacity	λ_k	Λ_k	Memory Capacity	λ_k	Λ_k
1	3.6GHz	0.5	0.73889	32GB	0.5	0.75
2	2.66GHz	0.5	0.90226	24GB	0.5	0.333
3	2.4GHz	N/A	N/A	8GB	N/A	N/A

It is important to note that the HEEP algorithm was designed for rack-scale composable DCs based on the discussions in earlier sections of this

paper. However, the heuristic can be extended to support a pod-scale composable DC should the need arise. This can be achieved by revising the workload blocking criterion in the heuristic to consider availability of all suitable resource component types at the pod-level rather than at the rack-level as described above.

4.6.2 Complexity Analysis

The MILP model is computationally intractable especially when larger number on binary variable are required. This is the case for in the MILP formulations in this chapter. The computational complexity of HEEP algorithm is $O(n)$. Where $n = (w \cdot p \cdot r \cdot s \cdot c)$; w is the number of workloads to be placed; p is the number of pods in the DC; r is the highest number of racks in any pod; s is the highest number of servers/nodes in any rack in the DC; and c is the maximum number of a unique resource component in a server/node. This is because in the worst case, to place all workloads energy efficiently in the composable DC, all resource components in the DC must be checked for each workload. However, effective use global knowledge in the algorithm is expected to ensure lower average execution time in a practical scenario. This is because global knowledge of DC infrastructure is obtained after placing the first workload. Such knowledge can be used to rapidly place some workloads without conducting a fresh search through all components in the DC.

4.6.3 Performance Evaluation

The performance of the HEEP algorithm is evaluated via a comparison with the optimal results obtained by solving the MILP model in preceding sections using similar input parameters. The results show that the total DC resource power consumption achieved by the HEEP algorithm replicates the trend reported from solving the MILP under different scenarios. Figure 4.27 and Figure 4.28 show the results obtained from MILP model and those obtained from the HEEP algorithm when 20 CPU and memory intensive are provisioned under different workload and DC architectures.

Similar to results obtained by solving the MILP model, the HEEP algorithm also shows that the highest total DC resource power consumption is observed when monolithic workloads are deployed in traditional DCs. Results obtained using the HEEP algorithm also shows the effectiveness of logical server disaggregation and adoption of more granular workload architecture towards the reduction of total DC power consumption. Relative to the MILP model results, Figure 4.27 shows that the highest percentage increase in the total DC resource power consumption observed when HEEP

algorithm is deployed for CPU intensive workloads under different DC and workload architectures is 14%. Under 20 memory intensive workloads, percentage increase in total DC resource power consumption of the HEEP algorithm relative to the results obtained from solving the MILP model does not exceed 11% for all scenarios in Figure 4.28.

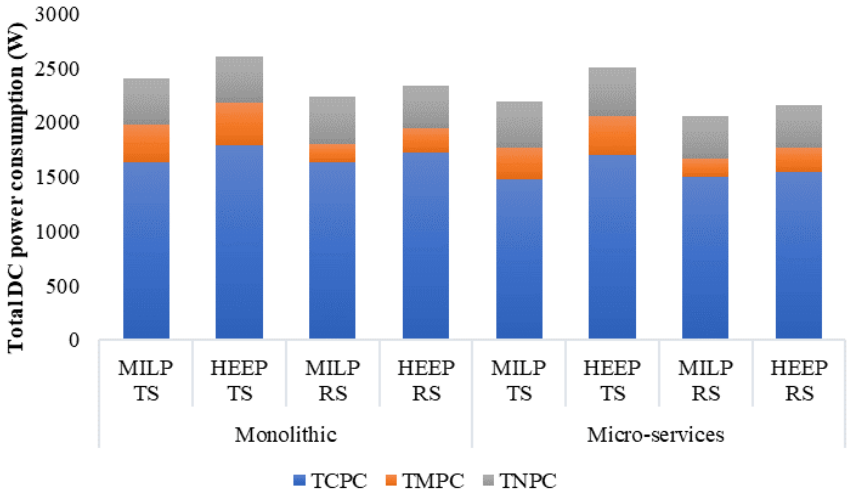


Figure 4.27: HEEP and MILP power consumption under 20 CPU intensive workloads

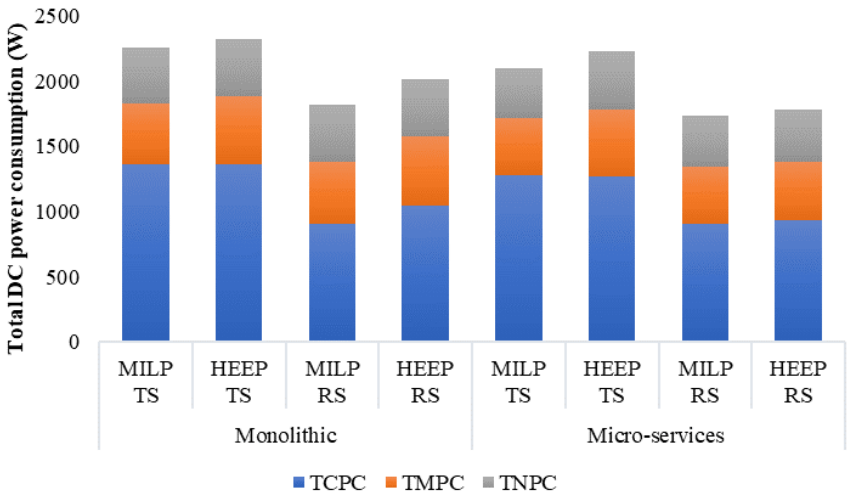


Figure 4.28: HEEP and MILP power consumption under 20 memory intensive workloads

Figure 4.29 and Figure 4.30 show the total DC power consumption when varying number of CPU and memory intensive monolithic workloads are used as input to both the MILP model and the HEEP algorithm. Figure 4.31 shows similar results for 20 integrated workloads comprising of micro-services. For the small number of in workloads (i.e. 5 - 20) considered, the results show that the HEEP algorithm is more effective for provisioning workloads in composable DCs than it is for traditional DCs. Relative to the MILP model, the

average percentage increase in the total DC power consumption when the HEEP algorithm is adopted in composable DCs are 6% and 8% for CPU and memory intensive workloads classes respectively. Compared to the MILP model, these values increase to 11% and 19% for CPU and memory intensive workload classes respectively when the HEEP algorithm is adopted in the traditional DC. Hence, the HEEP algorithm is somewhat less effective than the MILP in some scenarios. The greedy approach adopted by the HEEP algorithm is responsible for this. On the other hand, the MILP optimisation approach performs exhaustive search to obtain an exact solution. It is generally expected that the margins between results obtained using the MILP model and HEEP algorithm will decrease as the number and diversity of input workload demands in each workload class increase. This is because of the expected increase in the probability of workload consolidation onto DC resources. This will in turn lead to total DC resource power consumption that approach those reported from solving the MILP model.

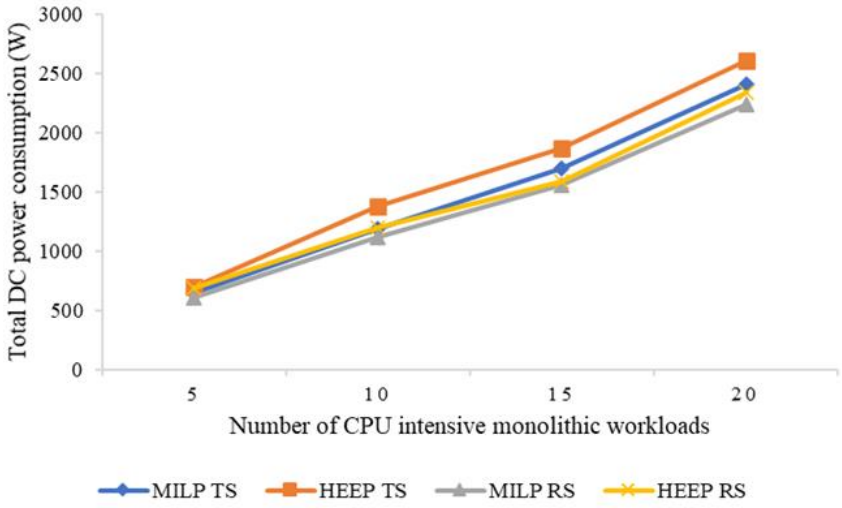


Figure 4.29: Comparison of HEEP and MILP power consumption under CPU intensive workloads

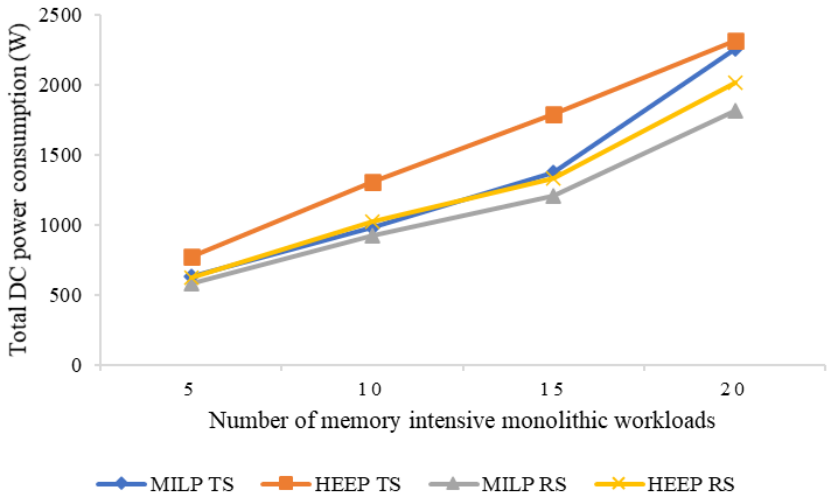


Figure 4.30: Comparison of HEEP and MILP power consumption under memory intensive workloads

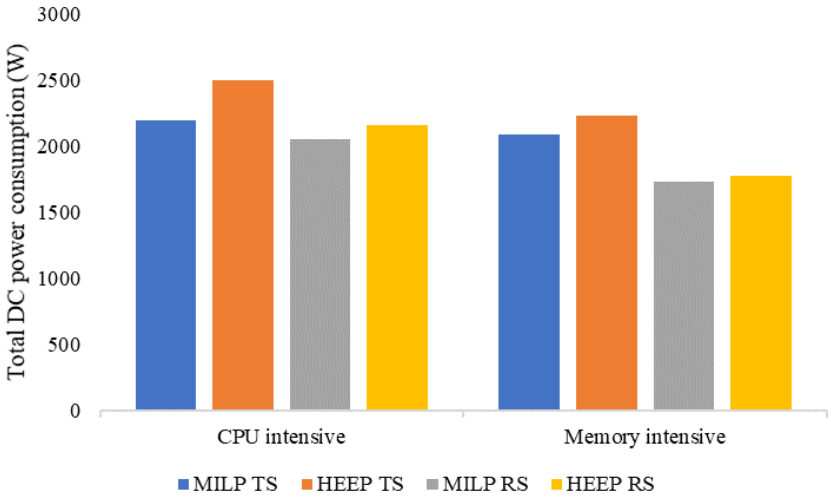


Figure 4.31: Comparison of HEEP and MILP model DC power consumption for 20 integrated workloads.

Furthermore, when executed on a basic personal computer with Intel Core i5-6400 6400 - 2.7 GHz Quad-Core Processor and 16 GB of memory, the execution time of the HEEP algorithm does not exceed 2 seconds in all evaluation scenarios. Hence, the HEEP algorithm is more practical and scalable for more realistic DC sizes relative to the adoption of a MILP model. Solving the MILP model for realistic DC sizes will require significantly larger computing power to obtain an exact solution. Future investigations will adopt the HEEP algorithm to evaluate the implementation of the composable infrastructure concept in realistic DC sizes. To improve the performance of both MILP model and heuristics, frequent re-optimisation of the MILP model based on changes in workloads distribution could be performed. Insights

derived from MILP model re-optimisation would guide re-optimisation of the heuristic for more specific scenarios as workload distribution changes.

4.7 Summary

In this chapter, we investigated the optimal scale and scope of resource disaggregation for an energy efficient composable DC infrastructure. We considered electrical, hybrid, and optical network topologies from the literature in the evaluation scenario. We developed a MILP model that places workloads energy efficiently in composable DCs while minimising both compute and network power consumption and workload rejection. By placing CPU intensive and memory intensive workloads energy efficiently in the infrastructure setup, our results showed that physical disaggregation at rack-scale is sufficient to achieve optimal utilisation of compute resources when resource configuration/allocation is suitable. Furthermore, our results also showed that the optical DC network is most suitable to achieve optimal energy efficiency in composable DCs. Relative to the traditional DC, physical disaggregation of server resources at rack-scale enable 5-8% and 6-20% saving in the total DC power consumptions when CPU and memory intensive workloads were considered respectively. In contrast to physical disaggregation of traditional servers at rack-scale, we found that logical disaggregation of traditional server over an optical network at rack-scale achieved comparable compute energy efficiency. Logical disaggregation of traditional server also enabled greater network energy efficiency in composable DCs. Furthermore, this chapter also studied the impact of increased workload modularity in composable DCs. Our result showed that increased workload modularity enable by the micro-service architecture complements increased resource modularity which resource disaggregation enables. Hence, both techniques should be combined to achieve optimal energy efficiency in composable DCs. Finally, we proposed a real-time heuristic for energy efficient placement of workloads in composable DCs. The heuristic replicated the trends and results produced by solving the MILP model.

Chapter 5 : Network Topologies for Composable Data Centres

5.1 Introduction

In this Chapter, we describe two variants of converged and targeted network for rack-scale composable DCs. We compare the performance of the targeted networks to that of another topology from the literature that adopts a generic design approach by formulating a MILP model. The MILP model conducts network load test in multi-rack set-up. Additionally, we conduct further evaluation of the novel topologies in realistic scenarios via an extended MILP model. The extended MILP model performs energy efficient placement of virtual machines in composable DCs that deploy the proposed network topologies. The composable DCs concurrently implement logical, hybrid or physical disaggregation at rack-scale.

5.2 Motivation

A suitable network topology is required to implement disaggregation in a composable infrastructure. Such network topology must support low latency and high bandwidth physical or logical connections between disaggregated resource components which exchange inter-resource traffic. Although, traditional DC network topologies may be maintained in composable DCs to operate in parallel with a dedicated topology that supports inter-resource data exchange, a converged network topology maybe preferred. A converged network topology can convey inter-resource traffic, east-west traffic, and north-south traffic concurrently. Hence, it may reduce network complexity and cost relative to dual topology setup. A converged network can also enable greater efficiency by improving network utilisation because both small (mice) and large (elephant) flows can be multiplexed into common network links.

Silicon photonics transceivers are widely used in today's DC. Optical fibre is also often used to interconnect servers in DCs. Further adoption of optical communication components, technologies, and techniques in DC environments can aid the emergence of a suitable network topologies for composable DC infrastructures. In Chapter 4, a reference all-optical composable DC network (EVROS) proved to be more energy efficient when compared to electrical and hybrid networks considered. The result was obtained in evaluation scenarios that assumed that all composable DC networks are un-capacitated. Un-capacitated networks can support the all resulting traffic that follows energy efficient placement of workload demands

into compute resources. Hence, network constraints were not considered in spite of the considering network cost as represented by power consumption. In practice, network constraints are extremely important factors that will determine the optimal performance of composable DCs. Regardless of the enormous potential bandwidth and low latency communication that the adoption of optical technology in the DC environment promises, attainable capacity remains limited by the following optical network constraints:

- the maximum data rate of single wavelength,
- number of interfaces/transceivers that can be packed onto single node,
- limited optical buffering capability, and
- lower efficiency associated with all-optical routing with limited or no wavelength/OEO conversions.

These limitations inhibit practical implementation of the reference all-optical composable DC network (EVROS) adopted in Chapter 4. For instance, adoption of all-optical routing without wavelength conversion in the inter-rack network would limit optimal utilisation of the network. Additionally, the deployment of EVROS in a practical DC, where each rack comprises of up to 48 resource nodes, is somewhat technically and financially challenging. This is because a practical implementation of the generic design approach adopted by EVROS as proposed requires that each resource node must support 48 interfaces to achieve full mesh physical connectivity within a rack.

Poor utilisation of inter-rack fabric of the EVROS topology was addressed in [89] where the authors proposed an enhanced variant called an all-optical programmable disaggregated DC network (AOPD-DCN). AOPD-DCN introduced architecture-on-demand (AoD) switches as shown in Figure 5.1. The AoD switches are deployed to replace the WSS top of cluster (ToC) switches and the inter-cluster switch. Each AoD switch comprise of both OCS and OPS modules. However, the challenge associated with the number of interfaces, which must be integrated onto each node, persists and it poses both technical and financial difficulties in a composable DC.

The view that high data rate links are required concurrently between all co-rack compute nodes in a disaggregated DC as proposed in EVROS and AOPD-DCN can be very costly and wasteful. For instance, if a traditional server with one CPU, one memory, one storage device and one NIC is physically disaggregated into four homogenous compute nodes. Given,

- CPU-to-memory traffic of 400 Gbps and 200 Gbps in the uplink and downlink directions respectively;
- CPU-to-storage traffic of 60 Gbps and 40 Gbps in the uplink and downlink directions respectively; and

- CPU-to-IO traffic of 10 Gbps and 8 Gbps in the uplink and downlink directions respectively.

Equation (5.1) shows the corresponding traffic matrix between the disaggregated compute components. The traffic matrix shows that the compute node with CPU is a hotspot which also requires high capacity interfaces. Similarly, the compute node with RAM also requires high capacity interfaces because of the high-bandwidth communication with the remote CPU. Capacity requirement of compute nodes with HDD and NIC require low-medium interface capacity. Furthermore, in rack with multiple disaggregated servers, it is unlikely that a compute node in a rack would communicate with all other co-rack compute nodes concurrently. Additionally, when a compute node holds multiple CPU components, i.e., a hotspot, it is unlikely in a practical scenario that such a node communicates with all other co-rack nodes at maximum capacity concurrently. In this chapter, a novel and more practical **Net**work for **Co**mposable **D**Cs (NetCoD) architecture is proposed. NetCoD addresses the limitation of the approach adopted in EVROS and AOPD-DCN.

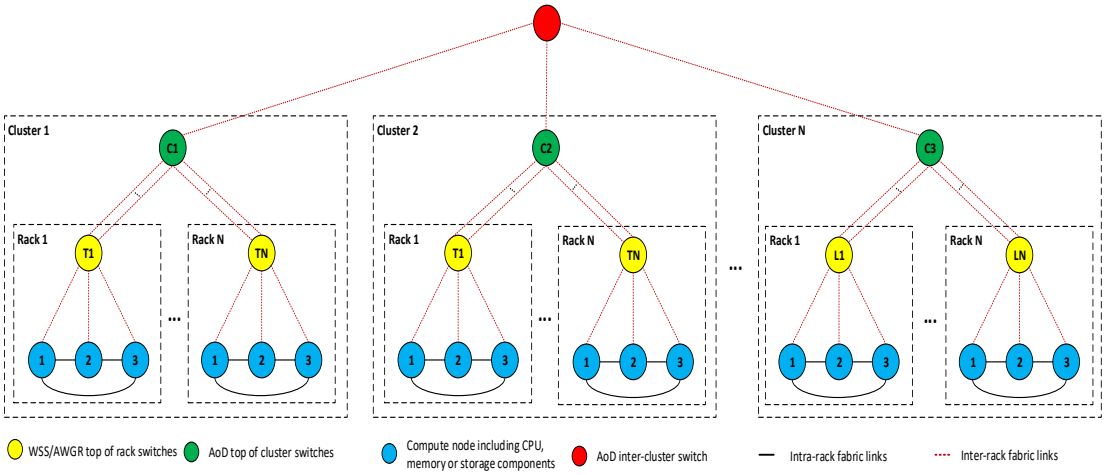


Figure 5.1: All-optical programmable disaggregated DC network(AOPD-DCN)

$$Traffic\ Matrix = \begin{bmatrix} \boldsymbol{Node} & CPU & RAM & HDD & NIC \\ CPU & 0 & 400 & 60 & 10 \\ RAM & 200 & 0 & 0 & 0 \\ HDD & 40 & 0 & 0 & 0 \\ NIC & 8 & 0 & 0 & 0 \end{bmatrix} \quad (5.1)$$

5.3 Network for Composable DC

Network for **Co**mposable **D**C (NetCoD) is a converged network topology that leverages optical communication technologies and silicon photonics. Consequently, it supports high-speed and low latency communication in

composable DCs. Two variants of NetCoD are described in this chapter i.e., electrical and electrical-optical variants. Both variants of NetCoD are designed for a composable DC that implements resource disaggregation at rack-scale. Therefore, inter-resource communication is limited to the internal network of each rack. On the other hand, traditional DC traffic i.e., east-west and north-south traffic traverses the inter-rack network of the DC. A common intra-rack network design is adopted in both variants of NetCoD. However, each variant integrates with a different inter-rack network in composable DCs.

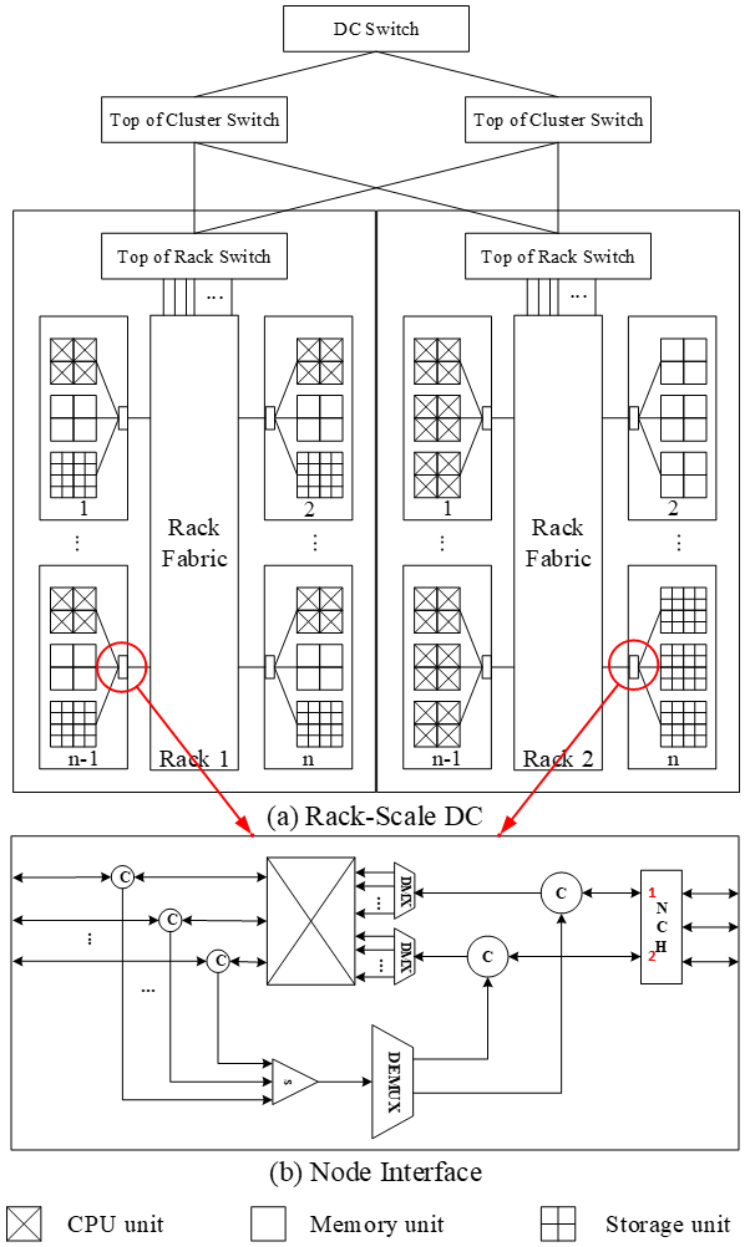


Figure 5.2: Network for composable DCs (NetCoD)

5.3.1 Intra-Rack Network

The intra-rack network within each rack, as shown in Figure 5.2, leverages optical communication components, technologies and techniques. The intra-

rack network supports high-speed and low latency communication between intra-rack resource components. Optical components such as optical backplane, optical circulators, combiners, demultiplexers, optical switches and silicon photonic transceivers are adopted at each compute node. The functions of the intra-rack fabric components are as follows:

- **Passive optical backplane:** The optical backplane is a passive wavelength routing network within each rack. It supports full mesh physical connectivity between nodes in the rack via point-to-point links. To minimise the size of the optical backplane within each rack due to unidirectional transmission on an optical link, bi-directional transmission may be employed. However, it is required that the same wavelength is not active in forward and reverse directions simultaneously when bi-directional communication is used. Wavelength division multiplexing (WDM) enables increased transmission capacity over a point-to-point optical link between two compute nodes. Furthermore, space division multiplexing (SDM) enables wavelength reuse on the optical backplane within the same rack. This is possible because each optical link establishes a dedicated point-to-point communication link between unique node pairs in the rack

- **Node Controller Hub:** Each node in the rack-scale composable DC infrastructure has a node controller hub (NCH). The NCH is proposed to replace the platform controller hub of traditional servers. As shown in Figure 5.2, all resource-components in a node are connected to the NCH. Resource components in the same node also maintain direct connectivity to one another via the node's on-board fabric to reduce the workload on the NCH, to ensure path diversity within the node and for greater energy efficiency. The NCH is a network element which performs network related computation in NetCoD. It may be implemented on an application-specific integrated circuit (ASIC) in commercial deployment and by a field-programmable gate array (FPGA) in experimental scenarios. The NCH performs the following functions:

    a. End-to-end virtual network setup (i.e., the routing function) for inter-nodal communication via direct or indirect physical links.
    b. Assignment of wavelengths for hop-to-hop communication (i.e., the forwarding function) over physical optical links.
    c. Multiplexing of data onto and the de-multiplexing of data from assigned inter-nodal wavelengths.
    d. Acting as an intermediate node on an indirect multi-hop path between two nodes.
    e. Optical switch path configuration to prevent wavelength collision on the passive optical backplane.
    f. Rate control and traffic scheduling as required to achieve optimal performance.

At each node, the NCH performs wavelength selection to avoid wavelength collision. Wavelength selection is performed based on global knowledge of the selection made at other nodes. Hence, all NCHs in NetCoD must be centrally controlled and orchestrated. This ensures optimal wavelength utilisation and the ability to operate NetCoD at maximum capacity.

- **Integrated Interfaces:** Integrated with each compute node's NCH are two interfaces. Each interface comprises of an array of optical transceivers that transmit and receive a set of pre-defined wavelengths. The wavelengths transmitted by one interface are received by the other interface and vice versa. This enables a node to use all the wavelengths supported by its interface for transmission and reception of data concurrently. A common interface pair is deployed in all compute nodes within each rack to enable easy replication and to leverage the benefits of economies of scale. The interface setup at each node promotes wavelength reuse in each rack and minimises the number of unique wavelengths required within each rack. Additionally, adoption of the interface pair at each node also enables path diversity which improves the resilience and capacity of NetCoD. The integration of a node's NCH element and the pair of interfaces may be implemented as a co-packaged device with optical IO by leveraging silicon photonics technologies.

- **De-multiplexer:** In the transmitting direction, the de-multiplexers at each node separate the wavelengths transmitted from the interface into the appropriate port of the optical switch. On the other hand, in the receiving direction, the de-multiplexer receives multiplexed wavelengths directed to a corresponding node from the passive optical backplane. Afterwards, the de-multiplexer forwards each wavelength to the interface that should receive the wavelength. This is achieved via pre-configured physical connection between the de-multiplexer and the pair of interfaces attached to each node.

- **Optical switches:** These are positioned before the point-to-point optical link between a node and the optical backplane. The optical switches prevent wavelength collision on the optical backplane and at the receiving nodes. Path configuration via the optical switch should be performed by NCH based on global knowledge. An integrated and energy efficient SOA-based optical switch with low switching speed is proposed to implement the optical switch.

- **Combiners**: In the receiving direction, the combiner at each compute node receives all wavelengths that have successfully traversed the optical backplane to reach the corresponding compute node. The combiner combines and forwards the received wavelengths to the de-multiplexer.

- **Optical Circulators**: Circulators enable bi-directional communication on optical links of the intra-rack backplane. Circulators are optional and may be employed between the optical backplane and the optical switch and also between the de-multiplexers and the integrated interfaces of each compute node. Adoption of bi-directional communication can reduce the size of each rack's optical backplane by half relative to the use of unidirectional communication. However, use of bidirectional communication over an optical link in the optical backplane may limit the attainable capacity because wavelength utilisation efficiency may reduce. It is important to note that the sets of transmitting and receiving wavelengths on each optical link must be mutually exclusive. This minimises the impact of crosstalk noise in the system when bi-directional transmission is employed over optical links.

5.3.1.1 Link Setup Process in Intra-Rack Network

The following process is implemented to setup a link between two nodes within a rack that employs NetCoD. The NCH selects wavelengths from the pool of wavelengths available at the pair of interfaces at each source node. The selected wavelengths must ensure collision free transmission on the optical backplane and at the destination node. In the transmitting direction, the wavelengths transmitted by the interfaces of each node flow through optical circulators to the de-multiplexer. The de-multiplexer separates all transmitted wavelengths of each node. and is connected to an optical switch. The optical switch directs the transmitted wavelengths to the appropriate link on the rack's optical backplane. Wavelength collision is avoided via the configuration of optical switches and via the use of parallel paths on the optical backplane. Therefore, dedicated communication paths can be established between each communicating nodes pair.

In the receiving direction, a combiner receives all transmitted wavelengths from other co-rack nodes and forwards the received wavelengths to a de-multiplexer. The de-multiplexer separates and forwards each received wavelength to the corresponding circulator that leads to the receiving interface. At the interface, each transceiver receives its associated wavelength and forwards the received data to the NCH. The NCH de-multiplexes the received data stream and forwards to the appropriate resource-component if it is in the destination node. Otherwise, the NCH forwards the received data to the corresponding interface linked to the next hop on the multi-hop communication path and selects an appropriate wavelength(s).

On the one hand, optical switches ensure that a wavelength is only transmitted to an intended destination node via the optical backplane. On the

other hand, combiners receive the ingress traffic (on the selected wavelengths destined for each node) from the optical backplane. Consequently, optical switches and combiners collectively reduce the number of interfaces required for each node to communicate over the full mesh optical backplane in a rack. This is because concurrent all-to-all communication is not expected between all co-rack nodes.

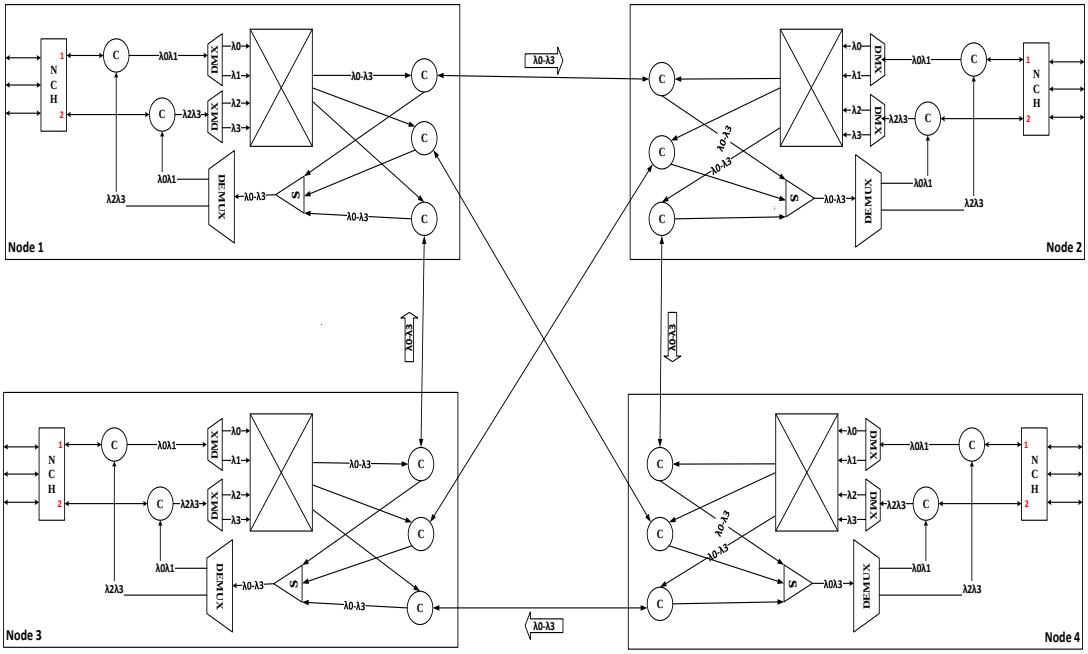


Figure 5.3: Wavelength assignment between 4 intra-rack nodes of NetCoD

As an illustration, consider a rack comprising of 4 compute nodes as illustrated in Figure 5.3. “Interface 1” of each NCH emits wavelengths λ0 and λ1 and receives λ2 and λ3 while “Interface 2” of each NCH emits wavelengths λ2 and λ3 and receives wavelengths λ0 and λ1. Figure 5.3 shows a at each node that leads to maximum throughput in the intra-rack network. This wavelength assignment does not violate network constraints under unidirectional or bi-directional transmission mode on optical links. As illustrated in Figure 5.3, Node 1 transmits wavelengths λ0-λ3 to Node 2; Node 2 transmits wavelengths λ0-λ3 to Node 4; Node 4 transmits wavelengths λ0-λ3 to Node 1; and Node 3 transmits wavelengths λ0-λ3 to Node 1. Hence, all nodes transmit and receive at full capacity by leveraging on WDM. Furthermore, SDM enables wavelength reuse on disjoint physical links as shown in Figure 5.3. It is important to note that the wavelength routing and assignment illustrated in Figure 5.3 is a solution to a MILP model that maximises throughput between four intra-rack nodes. Section 5.4.1 gives a full description of the MILP model that was solved.

5.3.2 Inter-Rack Network

Two variants of inter-rack network are proposed for NetCoD. The first variant called electrical-NetCoD (E-NetCoD) adopts a purely electrical inter-rack network because it comprises of only electrical switches. The second variant called electrical-optical-NetCoD (EO-NetCoD) adopts a hybrid inter-rack network which includes both electrical and optical switches. The physical topology depicted in Figure 5.4 is adopted for both variants of NetCoD in multi-cluster composable DC.

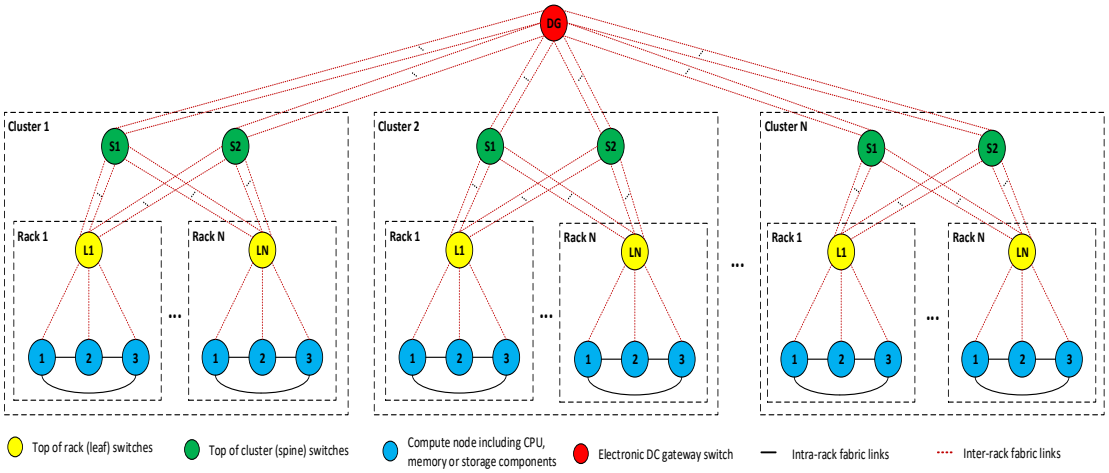


Figure 5.4: NetCoD in multi-cluster DC

5.3.2.1 Electrical NetCoD

In the electrical variant of NetCoD (E-NetCoD), the optical backplane of the intra-rack network also includes additional point-to-point links. These additional links connect compute nodes in each rack to a bespoke electrical leaf switch that functions as a top of rack (ToR) switch. The intra-rack network integrates with an electrical leaf-spine DCN topology via such links a shown in Figure 5.4. The leaf switches are equipped with specialised interfaces to enable communication with compute nodes within the same rack via the NCH. The bespoke leaf switch within each rack and NCH (attached to each compute node in the rack) are centrally orchestrated to avoid wavelength collision. It is assumed that the bespoke leaf switch can perform wavelength conversion as required. The leaf switch is also expected to possess intrinsic intelligence to select wavelengths that avoid collision when communicating with each NCH. The leaf switch in each rack connects to the electrical spine (ToC) switches in the higher tier of the leaf–spine DCN topology. The spine switches connect to electrical super-spine/gateway switch to support inter-cluster communication and north-south communication in the composable DC.

All electrical switches in the topologies perform routing and forwarding functions. However, the super-spine switch is expected to support higher capacity relative to other electrical switches in the leaf-spine physical topology. A leaf-spine network topology is employed in the inter-rack fabric because of its well-known advantages such as robustness enabled via path diversity and non-disruptive scalability. Additionally, the used of multiple aggregated physical links between switches in the inter-rack fabric may be implemented in large deployment scenarios, as shown in Figure 5.4, to improve capacity as required. The integration of intra-rack fabric and inter-rack leaf-spine topology of E-NetCoD conveys all traffic types (i.e. inter-resource traffic, east-west traffic, and north-south traffic) in the composable DC. However, rack-scale disaggregation ensures that inter-resource traffic is limited to each rack. This prevents oversubscription and throughput challenges that may otherwise arise in such a converged network that is deployed in a composable DC.

In addition to supporting inter-rack traffic exchanges, the leaf (ToR) switches can also function as an intermediate node for inter-resource traffic and east-west traffic exchange within the same rack. A low latency electrical switch such as the switch proposed by the Gen-Z consortium [82] may be adopted as the leaf (ToR) switch. The integration of the optical backplane and the leaf-spine topology in each rack enables additional paths for inter-resource communication within the rack. Therefore, improving capacity and robustness. However, network bottlenecks resulting from the adoption of a shared medium for all communication types may occur at each node. It is expected that higher capacity of single wavelength data rate in optical links will mitigate such bottlenecks. In recent times, up to 100 Gbps single wavelength transmission have been demonstrated [107] and even high capacity is expected as optical technologies advance. Notwithstanding, a complex control mechanism is required to effectively manage the transmission of heterogeneous traffic types concurrently on the same media.

5.3.2.2 Electrical-Optical NetCoD

The electrical-optical variant of NetCoD (EO-NetCoD) adopts optical switches in place of electrical leaf and spine switches of the E-NetCoD while maintaining an electrical gateway switch. This reduces the OEO conversions in the network topology to enable reduced latency and reduced power consumption. This is because each compute node can select appropriate wavelengths to establish both intra-rack and inter-rack light-paths. Alternatively, multiple light-paths may be established to facilitate inter-rack

communication via intermediate nodes. The electrical gateway switch or compute nodes in other racks of the composable DC can act as an intermediate node. It is assumed that the high capacity electrical gateway switch can perform wavelength conversion as required. It is also assumed that the gateway switch has intrinsic intelligence to select wavelengths that avoid collision when communicating over the inter-rack network.

A limitation of EO-NetCoD is the degradation in network performance resulting from wavelength continuity when routing is performed solely in the optical domain with limited wavelength and OEO conversions. Wavelength continuity leads to reduction in wavelength utilisation and higher number of network connection request rejections in optical networks. Hence, it reduces network flexibility relative to a network that performs more OEO or wavelength conversions. However, this challenge may be mitigated when high single wavelength transmission rate is adopted. Higher single wavelength transmission rate is constrained by technological advancement. Hence, adoption of greater path diversity between switches in the inter-rack network is proposed to further mitigate challenges introduced by wavelength continuity as given in Figure 5.4. Factors that may determine the number of diverse paths provisioned between switches of the inter-rack network include but are not limited to DC cluster size, size of the wavelength-pool supported in the NetCoD system and network availability criteria desired in the DC. It is expected that the adoption of rack-scale disaggregation will enable significant reductions in the volume of inter-rack traffic in the composable DC. This is because only east-west traffic and north-south traffic types will traverse inter-rack network. Such traffic types have relatively lower bandwidth and higher tolerance to latency. Furthermore, a strategy that groups and places workloads with frequent east-west traffic exchange inside the same rack will also reduce the east-west traffic between racks in the composable DC.

In contrast to E-NetCoD where electrical switches enable wavelength and OEO conversions intrinsically, the adoption of optical switches in EO-NetCoD increases the likelihood of wavelength collision. This consequently reduces wavelength reuse opportunities. On the one hand, NCH element attached to each compute node can enable wavelength and OEO conversions. This can be achieved via selection of appropriate wavelengths for hop-to-hop communication over both intra-rack and inter-rack networks. Furthermore, the electrical gateway switch forms an important boundary for wavelength collision and reuse in the inter-rack network of EO-NetCoD. To complement similar functions performed by the NCH, the gateway switch also

performs OEO and wavelength conversions. The boundary introduced by the gateway switch limits the wavelength collision domain to each cluster of the composable DC. Hence, each cluster is an independent wavelength collision domain. The pool of supported wavelengths can be independently reused in each cluster of the composable DC to maximise wavelength utilisation and total network capacity. Additionally, the capacity of links between optical switches in the inter-rack network is limited because wavelength collision avoidance is required in the all-optical layer. To overcome this limitation, path diversity should be employed between switches in the inter-rack network to improve capacity in large deployment scenarios.

It is important to note that a passive all-optical switch such as the arrayed waveguide grating router (AWGR) may be used to implement the leaf-spine layer of the EO-NetCoD. However, there are inherent disadvantages of using a passive optical switch which has a fixed routing matrix. Such design reduces cost efficiency and energy efficiency because multiple transceivers must be fitted onto each node's interface. For example, consider a single rack with 48 servers. To achieve full mesh connectivity between all servers in that rack via single hop communication through an AWGR (without the use of time slots on wavelengths), a 48x48 AWGR is required. Each node's interface must support the transmission and reception of 48 unique wavelengths. On the other hand, multi-hop communication path can be used to achieve virtual full mesh connectivity. However, this implies that routing and forwarding costs (power consumption) must be incurred at intermediate nodes on the communication path. In scenarios with high multi-hop communication, such power consumptions may outweigh any power savings achieved via the adoption of a passive optical switch with zero power consumption. Therefore, a configurable optical switch such as an optical cross connect (OXC) is proposed for EO-NetCoD to enable an adaptable and dynamic network for composable DCs.

5.3.3 Scaling in NetCoD

At the rack level, both variants of NetCoD scale-out via incremental and non-disruptive installation of additional compute nodes. The newly added compute nodes are connected to existing compute nodes and to the ToR switch in that rack via the passive optical backplane. At the cluster level, NetCoD scales-out via incremental and non-disruptive installation of more racks. The additional racks are connected to the dedicated leaf-spine inter-rack network of each cluster. Finally, at the DC-level, NetCoD supports on-demand scale-out via

incremental and non-disruptive installation of more clusters which are connected to the electrical gateway switch of the composable DC.

5.4 MILP Model for Network Topology for Composable DCs

This section presents a MILP model that optimises both variants of NetCoD. The MILP model is also revised to implement AOPD-DCN. The MILP model performs routing and forwarding of network traffic over the corresponding network topology to minimise or maximise a specific objective.

5.4.1 MILP Model for E-NetCoD

The model sets, parameters, and variables for a composable DC that implements E-NetCoD are given as follows.

Sets:

A	Set of compute nodes $A \subseteq N$
G	Set of DC gateway switches to the Internet $G \subseteq N$
Y	Set of compute nodes and DC gateway switches $Y \subseteq N$; $Y = A \cup G$
Z	Set of leaf and spine switches $Z \subseteq N$
Q	Set of routing and forwarding nodes in the DC, $Q \subseteq N$; $Q = Z \cup A \cup G$
N	Set of all Nodes, $N = Z \cup A \cup G$
N_m	Set of all neighbour nodes of node $m \in N$; $N_m \subseteq N$.
B_m	Set of all intra-rack neighbour nodes of node $m \in N$; $B_m \subseteq N$.
A_m	Set of all compute nodes that are neighbours of compute node $m \in A$; $A_m \subseteq A$.
O	Set of transmission wavelengths supported in the network.
T	Set of interfaces supported by a compute node.

Parameters:

$\mathbb{T}_{of}$	$\mathbb{T}_{of} = 1$ if wavelength $o \in O$ is allocated to interface $f \in T$ for transmission of data traffic, otherwise $\mathbb{T}_{of} = 0$
$\mathbb{R}_{of}$	$\mathbb{R}_{of} = 1$ if wavelength $o \in O$ is allocated to interface $f \in T$ for reception of data traffic, otherwise $\mathbb{R}_{of} = 0$

μ_i	Load proportional routing cost for a routing and forwarding node $i \in Q$, (J/b)
OB	On-board network interface energy per bit (J/b)
TX_m	Transmitting energy per bit (J/b) of routing and forwarding node $m \in Q$
RX_m	Receiving energy per bit (J/b) of routing and forwarding node $m \in Q$
RL_m	Relaying energy per bit (J/b) of routing and forwarding node $m \in Q$
$\mathcal{X}$	Optical switch operational power in Watt
$\mathcal{E}$	Electrical switch operational power in Watt
$\mathcal{S}$	SOA switch energy per bit (J/b)
ρ	Number of spine switches in the composable DC
ϱ	Number of electrical gateway or super-spine switches in the composable DC
$\mathbb{r}$	Number of active racks in the composable DC
τ_{sd}	Total traffic from node $s \in Q$ to node $d \in Q$.
$\mathcal{D}$	Maximum data rate of a single wavelength.
$\mathcal{Q}$	A big number (100000)
$\mathcal{G}$	A big number (1000)

Variables

$\mathbf{T_{sd}}$	Total traffic from node $s \in \mathrm{Q}$ to node $d \in \mathrm{Q}$.
$\mathbf{T_{sd}^{ij}}$	Volume of T_{sd} traversing virtual link (i,j). $i \in \mathrm{Q}, j \in Q, s \in \mathrm{Q}, d \in \mathrm{Q}: i \neq j, s \neq d$. It denotes routing of traffic in the virtual network.
$\boldsymbol{\upsilon_{ij}}$	Volume of traffic on virtual link (i,j); $i \in Q, j \in \mathrm{Q}$
$\boldsymbol{\Phi_i}$	Traffic transmitted at routing node $i \in Q$
$\boldsymbol{\Psi_i}$	Traffic received at routing node $i \in Q$
$\boldsymbol{\Omega_i}$	Traffic relayed at routing node $i \in Q$
$\boldsymbol{\phi_m}$	Traffic transmitted at forwarding node $m \in Q$
$\boldsymbol{\psi_m}$	Traffic received at forwarding node $m \in Q$

$\boldsymbol{\omega_m}$ Traffic relayed at forwarding node $m \in Q$

$\boldsymbol{v_{omn}^{ij}}$ Volume of traffic on virtual link (i,j) using wavelength $o \in O$ on physical link $(m,n), i \in Q, j \in Q, m \in N, n \in N_m: i \neq j, m \neq n$

$\boldsymbol{w_{omn}}$ Volume of traffic using wavelength $o \in O$ on physical link $(m,n),\ m \in N, n \in N_m: m \neq n$

$\boldsymbol{q_{omn}}$ $\boldsymbol{q_{omn}} = 1$ if $\boldsymbol{w_{omn}} > 0$. Otherwise $\boldsymbol{q_{omn}} = 0, o \in O, m \in N, n \in B_m: m \neq n$

$\boldsymbol{g_{omn}}$ $\boldsymbol{g_{omn}} = 1$, if $\boldsymbol{q_{omn}} \vee \boldsymbol{q_{onm}} = 1$. Otherwise $\boldsymbol{g_{omn}} = 0, o \in O, m \in N, n \in B_m: m \neq n$

$\boldsymbol{e_{ofm}}$ $\boldsymbol{e_{ofm}} = 1$ if wavelength $o \in O$ is used on interface $f \in T$ of compute node $m \in A$ either to transmit traffic to neighbour nodes or receive traffic from neighbour nodes. Otherwise, $\boldsymbol{e_{ofm}} = 0$.

The variables are related as follows.

$$\boldsymbol{w_{omn}} \geq \boldsymbol{q_{omn}} \quad \forall o \in O, \forall m \in N, n \in B_m: m \neq n \tag{5.2}$$

$$\boldsymbol{w_{omn}} \leq Q\, \boldsymbol{q_{omn}} \quad \forall o \in O, \forall m \in N, n \in B_m: m \neq n \tag{5.3}$$

Equations (5.2) and (5.3) derive the state of each wavelength available on each physical link within the rack.

$$\boldsymbol{\Phi_i} = \sum_{j \in Q} \sum_{d \in Q} \mathbf{T}_{\boldsymbol{id}}^{\boldsymbol{ij}} \quad \forall i \in Q: d \neq i, i \neq j \tag{5.4}$$

Equation (5.4) derives the traffic transmitted by a routing and forwarding node in the virtual layer of the network topology.

$$\boldsymbol{\Psi_i} = \sum_{s \in \mathrm{Q}} \sum_{j \in \mathrm{Q}} \mathbf{T}_{\boldsymbol{si}}^{\boldsymbol{ji}} \quad \forall i \in \mathrm{Q}: s \neq i, i \neq j \tag{5.5}$$

Equation (5.5) derives the traffic received by a routing and forwarding node in the virtual layer of the network topology.

$$\boldsymbol{\Omega_i} = \sum_{s \in Q} \sum_{d \in Q} \sum_{j \in Q} \mathbf{T}_{\boldsymbol{sd}}^{\boldsymbol{ij}} \tag{5.6}$$

$$\forall i \in Q: s \neq d, s \neq i, d \neq i, i \neq j$$

Equation (5.6) derives the traffic relayed by a routing and forwarding node in the virtual layer of the network topology.

$$\boldsymbol{\phi_m} = \sum_{o \in O} \sum_{n \in N_m} \sum_{j \in Q} \boldsymbol{v_{omn}^{mj}} \tag{5.7}$$

$$\forall m \in Q: j \neq m, m \neq n$$

Equation (5.7) derives the traffic transmitted by a routing and forwarding node in the physical layer of the network topology.

$$\boldsymbol{\psi_m} = \sum_{o \in O} \sum_{n \in N_m} \sum_{i \in Q} \boldsymbol{v_{onm}^{im}} \tag{5.8}$$

$$\forall m \in Q: i \neq m, m \neq n$$

Equation (5.8) derives the traffic received by a routing and forwarding node in the physical layer of the network topology.

$$\boldsymbol{\omega_m} = \sum_{o \in O} \sum_{n \in N_m} \sum_{j \in Q} \sum_{i \in Q} \boldsymbol{v_{wmn}^{ij}} \tag{5.9}$$

$$\forall m \in Q: i \neq j, i \neq m, j \neq m, m \neq n$$

Equation (5.9) derives the traffic relayed by a routing and forwarding node in the physical layer of the network topology.

$$\boldsymbol{TNRP} = \mu_i \sum_{i \in Q} (\boldsymbol{\Phi_i} + \boldsymbol{\Psi_i} + \boldsymbol{\Omega_i}) \tag{5.10}$$

Equation (5.10) derives the total network routing power (TNRP) due to the routing function performed by routing and forwarding nodes in the virtual layer. This represents any additional power consumed by nodes that perform routing in the logical layer.

$$\boldsymbol{TNFP} = \sum_{i \in Q} (\boldsymbol{\phi_i}\, TX_i + \boldsymbol{\psi_i}\, RX_i + \boldsymbol{\omega_i}\, RL_i) \tag{5.11}$$

Equation (5.11) determines the total network forwarding power (TNFP) due to the forwarding function performed by routing and forwarding nodes in the physical layer.

$$\boldsymbol{TXNP} = \sum_{i \in A} (\boldsymbol{\phi_i} + \boldsymbol{\omega_i})\, \mathcal{S} + (\mathbb{r} + \rho + \varrho)\mathcal{E} \tag{5.12}$$

Equation (5.12) establishes the total other network power ($TXNP$) for E-NetCoD which is measured by the power consumed by active physical node/components in the network topology. It comprises of the power

consumed by SOA switches at compute nodes and the fixed operational power of electrical switches in the composable DC.

Total Network Power Consumption

$$\boldsymbol{TNPC = TNFP + TNRP + TXNP} \tag{5.13}$$

The total network power consumption ($\boldsymbol{TNPC}$) is a sum of the $\boldsymbol{TNFP}$, $\boldsymbol{TNRP}$ and $\boldsymbol{TXNP}$.

The model MILP is defined as follows:

Two distinct objective functions are considered for the model to represent two distinct scenarios.

Objective 1: Maximise:

$$\sum_{s \in Y} \sum_{d \in Y: s \neq d} \mathbf{T}_{\boldsymbol{sd}} \tag{5.14}$$

Equation (5.14) is the first objective function that maximises the total throughput between all desired communicating nodes pairs in the composable DC. This objective function maximises the total traffic exchanged between selected routing and forwarding nodes in the DC. Note that $\mathbf{T}_{\boldsymbol{sd}}$ is a variable determined by the model under this scenario.

Objective 2: Minimise:

$$\boldsymbol{TNPC} \tag{5.15}$$

Equation (5.15) is the second objective function. It minimises the total network power consumed by routing and forwarding the input traffic τ_{sd} over the corresponding network topology under consideration. Hence, τ_{sd} is an input parameter to the model in this scenario.

Subject to:

$$\sum_{j \in Q: i \neq j} \mathbf{T}_{\boldsymbol{sd}}^{\boldsymbol{ij}} - \sum_{j \in Q: i \neq j} \mathbf{T}_{\boldsymbol{sd}}^{\boldsymbol{ji}} = \begin{cases} \mathbf{T}_{\boldsymbol{sd}} & i = s \\ -\mathbf{T}_{\boldsymbol{sd}} & i = d \\ 0 & otherwise \end{cases} \tag{5.16}$$

$$\forall\, i \in Q, \forall\, s, d \in Y: s \neq d$$

Constraint (5.16) enforces flow conservation in the virtual layer setup between electronic routing and forwarding nodes in the composable DC. Note that for objective 2, $\mathbf{T}_{\boldsymbol{sd}}$ should be replaced by τ_{sd} in (5.16).

$$\sum_{s \in Y} \sum_{d \in Y: s \neq d} \mathbf{T}_{\boldsymbol{sd}}^{\boldsymbol{ij}} = \boldsymbol{\vartheta_{ij}} \tag{5.17}$$

$$\forall\, i \in Q, \forall\, j \in Q: i \neq j$$

Constraint (5.17) calculates the volume of traffic on each virtual link provisioned between a pair of routing and forwarding nodes in the virtual layer.

$$\sum_{o \in O} \sum_{n \in N_m : m \neq n} \boldsymbol{v}_{omn}^{ij} - \sum_{o \in O} \sum_{n \in N_m : m \neq n} \boldsymbol{v}_{onm}^{ij} = \begin{cases} \boldsymbol{v}_{ij} & m = i \\ -\boldsymbol{v}_{ij} & m = j \\ 0 & otherwise \end{cases} \quad (5.18)$$

$$\forall\, m \in N, \forall\, i,j \in Q : i \neq j$$

Constraint (5.18) enforces flow conservation in the physical network topology between all nodes in the DC.

$$\sum_{i \in Q} \sum_{j \in Q : i \neq j} \boldsymbol{v}_{omn}^{ij} = \boldsymbol{w}_{omn} \quad (5.19)$$

$$\forall\, o \in O, \forall\, m \in N, \forall\, n \in N_m : m \neq n$$

Constraint (5.19) calculates the volume of traffic on each wavelength on a physical link in the network topology.

$$\boldsymbol{w}_{omn} \leq \mathcal{D} \quad (5.20)$$

$$\forall\, o \in O, \forall\, m \in N, n \in N_m : m \neq n$$

Constraint (5.20) is the capacity constraint of each wavelength used on a physical link.

$$\sum_{n \in B_m} \sum_{f \in T} \boldsymbol{q}_{omn}\, \mathbb{T}_{of} \leq 1 \quad (5.21)$$

$$\forall\, o \in O, \forall\, m \in A : m \neq n$$

$$\sum_{n \in B_m} \sum_{f \in T} \boldsymbol{q}_{onm}\, \mathbb{R}_{of} \leq 1 \quad (5.22)$$

$$\forall\, o \in O, \forall\, m \in A : m \neq n$$

Constraint (5.21) ensures that each wavelength transmitted by a compute node is transmitted once from that node by an interface that is designed to emit that wavelength. On the other hand, Constraint (5.22) ensures that each wavelength received by a compute node is received once at an interface that is designed to receive that wavelength.

$$\sum_{n \in B_m : m \neq n} \boldsymbol{q}_{omn}\, \mathbb{T}_{of} + \sum_{n \in B_m : m \neq n} \boldsymbol{q}_{onm}\, \mathbb{R}_{of} = \boldsymbol{e}_{ofm} \quad (5.23)$$

$$\forall\, o \in O, \forall f \in T, \forall m \in A$$

Constraint (5.23) ensures that the same wavelength does not flow in opposite directions at a given interface of a compute node. This is required when bi-directional communication is employed on the physical link that connects each

interface to the optical backplane and each interface comprises of an array of unique transceivers.

$$\boldsymbol{q_{omn}} + \boldsymbol{q_{onm}} = \boldsymbol{g_{omn}} \tag{5.24}$$

$$\forall\, o \in O, \forall\, m \in A, n \in B_m : m \neq n$$

Constraint (5.24) ensures that same wavelength does not flow in opposite directions in a physical link on each rack's optical backplane. This constraint implements bi-directional communication on the passive intra-rack backplane. It is not active when unidirectional communication is implemented on the passive intra-rack backplane.

5.4.2 MILP Model for EO-NetCoD

In contrast to E-NetCoD, some network constraints must be revised to represent EO-NetCoD in a MILP model while others must be introduced. Consequently, additional set, parameter and variables are introduced while others are revised as given below. The additional set enable the representation of revised nodes when EO-NetCoD is implemented. The hybrid inter-rack network is created via the adoption of optical switches to replace electrical leaf and spine switches. The hybrid inter-rack network comprises of all nodes that are connected directly to any optical switch in the physical network topology. Such nodes include compute nodes, optical switches, and DC gateway switch. The additional variables enable representation of traffic routing over EO-NetCoD.

Revised and additional Sets and Parameter

Q	Set of all routing and forwarding nodes $Q = Y = A \cup G$
N	Set of all nodes $N = X \cup A \cup G$
N_m	Set of all neighbour nodes of node $m \in N, N_m \subseteq N$
X	Set of optical switches, $X \subseteq N$
H_m	Set of all neighbour nodes of node $m \in N$; $H_m \subseteq N$ which are part of the hybrid inter-rack network.
κ	Cost associated with each path provisioned in an optical switch in Watt

Additional variables

$\boldsymbol{u_{ij}}$	Volume of traffic on virtual link (i, j), $i \in Q, j \in Q$, that traverses intra-rack network.

$\boldsymbol{\ell_{oij}}$ Volume of traffic using wavelength $o \in O$ on virtual link (i,j), $i \in Q, j \in Q$, that traverses the hybrid inter-rack network.

$\boldsymbol{\mathcal{L}_{oij}}$ $\boldsymbol{\mathcal{L}_{oij}} = 1$ if $\boldsymbol{\ell_{oij}} > 0$. Otherwise, $\boldsymbol{\mathcal{L}_{oij}} = 0, o \in O, i \in Q, j \in Q: i \neq j$

$\boldsymbol{x_{omxn}}$ The configured switching matrix of an optical switch. $\boldsymbol{x_{omxn}} = 1$ if the wavelength $o \in O$ from node $m \in H_x$ enters optical switch $x \in X$ and is relayed to node $n \in H_x$. Otherwise, $\boldsymbol{x_{omxn}} = 0$.

$\boldsymbol{d^{ij}_{omxn}}$ $\boldsymbol{d^{ij}_{omxn}}$ gives the traffic $\boldsymbol{v^{ij}_{omx}}$ that enters optical switch $x \in X: x \in H_m$ from node $m \in H_x$ and is relayed to node $n \in H_x$ on the hybrid inter-rack network. $o \in O, m \in N$

$$\boldsymbol{d^{ij}_{omxn}} = \boldsymbol{v^{ij}_{omx}}\, \boldsymbol{x_{omxn}}$$

An illustration of the derivation of variable $\boldsymbol{d^{ij}_{omxn}}$ is given by the following example in Figure 5.5. Given a 4x4 optical switch with the configured switching matrix illustrated in Figure 5.5a, If node 1, transmits W Gbps on wavelength λ_1 to the optical switch, the traffic will be passively and transparently forwarded to node 4. Note that, node 1 is connected to the optical switch on port 1 and node 4 is connected to port 4 of the optical switch as illustrated in Figure 5.5b.

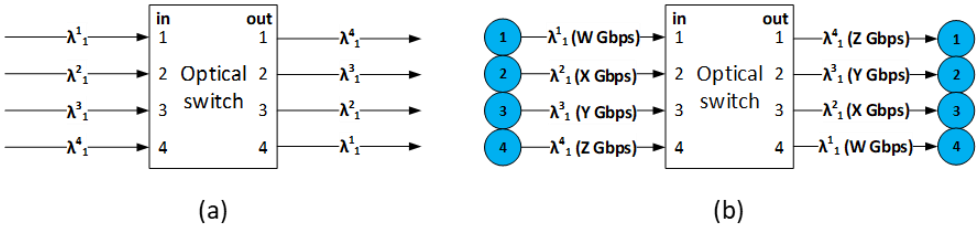


Figure 5.5:(a) Optical switch configuration ($\boldsymbol{x_{omxn}}$). (b) Traffic switching by optical switch ($\boldsymbol{d^{ij}_{omxn}}$).

The EO-NetCoD comprise of two intrinsic networks as illustrated in Figure 5.4. That is, an intra-rack network between routing and forwarding capable compute nodes within the same rack and a hybrid inter-rack network enabled by the deployment of optical switches in a leaf-spine topology. Point-to-point light-paths are setup between routing nodes over the physical links of the hybrid inter-rack network. After path configuration, the optical switches are in a passive state. Hence, the optical switches are only aware of directly connected neighbours on the hybrid inter-rack network.

The $\boldsymbol{TXNP}$ for EO-NetCoD can be measured by the power consumed by active physical node/components in the network topology. It comprises of the power consumed by SOA switches at compute nodes, the fixed operating power of optical and electrical switches and the cost of setting up optical paths in each optical switch in the composable DC. It is derived as by Equation 5.25.

$$\boldsymbol{TXNP} = \sum_{i \in A} (\boldsymbol{\phi_i} + \boldsymbol{\omega_i})\, \mathcal{S} + (\mathbb{r} + \rho)\, \mathcal{X} + \varrho\, \mathcal{E} + \kappa \left(\sum_{o \in O} \sum_{x \in X} \sum_{m \in H_x} \sum_{n \in H_x} \boldsymbol{x_{omxn}} \right) \tag{5.25}$$

Given the traffic demand $\mathbf{T_{sd}}$ in the DC, virtual links $\boldsymbol{v_{ij}}$ are setup between routing and forwarding nodes in the DC to efficiently route traffic over the network topology as seen earlier in Constraints (5.16) and (5.17). However, a virtual link $\boldsymbol{v_{ij}}$ between two routing and forwarding nodes in EO-NetCoD can traverse the intra-rack network and/or the hybrid inter-rack network. As a result, Constraint (5.18) is no longer applicable in such a setup. To accommodate such a unique setup in the MILP model, the following network constraints are introduced.

$$\boldsymbol{u_{ij}} + \sum_{o \in O} \boldsymbol{\ell_{oij}} = \boldsymbol{v_{ij}} \tag{5.26}$$

$$\forall\, i \in Q, \forall\, j \in Q : i \neq j$$

Constraint (5.26) ensures that the volume of traffic on virtual link v_{ij} is equal to the sum of traffic sent via the intra-rack network and the hybrid inter-rack network. This is because the virtual link can be routed via the intra-rack network or/and via the inter-rack network.

$$\sum_{o \in O} \sum_{n \in A_m : m \neq n} \boldsymbol{v_{omn}^{ij}} - \sum_{o \in O} \sum_{j \in A_m : m \neq n} \boldsymbol{v_{onm}^{ij}} = \begin{cases} \boldsymbol{u_{ij}} & m = i \\ -\boldsymbol{u_{ij}} & m = j \\ 0 & otherwise \end{cases} \tag{5.27}$$

$$\forall\, i, j \in Q : s \neq d\ \forall\, m \in A$$

Constraint (5.27) enforces flow conservation in physical links of the intra-rack network of each rack in the DC.

$$\sum_{n \in H_m : m \neq n} \boldsymbol{v_{omn}^{ij}} - \sum_{n \in H_m : m \neq n} \boldsymbol{v_{onm}^{ij}} = \begin{cases} \boldsymbol{\ell_{oij}} & m = i \\ -\boldsymbol{\ell_{oij}} & m = j \\ 0 & otherwise \end{cases} \tag{5.28}$$

$$\forall\, m \in N, \forall\, i, j \in Q : i \neq j, \forall\, w \in W$$

Constraint (5.28) enforces flow conservation in physical links of the hybrid inter-rack network in the DC. It also enforces wavelength continuity on each light-path created between two nodes in the inter-rack network.

$$\boldsymbol{\ell_{oij}} \geq \boldsymbol{\mathcal{L}_{oij}} \tag{5.29}$$

$$\forall\, o\ \in O, \forall\, i \in Q, j \in Q : i \neq j$$

$$\boldsymbol{\ell_{oij}} \leq \mathcal{Q}\, \boldsymbol{\mathcal{L}_{oij}} \tag{5.30}$$

$$\forall\, o\ \in O, \forall\, i\ \in Q, j \in Q : i \neq j$$

Constraints (5.29) and (5.30) jointly derive the state of all potential light-paths that can traverse the hybrid inter-rack network.

$$\sum_{j\,\in Q} \boldsymbol{\mathcal{L}_{oij}} \leq 1 \tag{5.31}$$

$$\forall\, o\ \in O, \forall\, i\ \in A : i \neq j$$

Constraint (5.31) ensures that a wavelength is used to setup a single light-path from a given compute node. Hence, the wavelength must not be used more than once on any light-path originating at this compute node. Note that the gateway switch is permitted to use a given wavelength on different light-paths provided that network routing constraints are not violated. This is because a sophisticated network switch is assumed. Multiple links emanate from the gateway switch i.e., path diversity is employed; hence, the risk of wavelength collision is mitigated since a wavelength can be re-used on disjoint links.

Optical network routing constraints

$$\sum_{n\,\in H_x} \boldsymbol{x_{omxn}} \leq 1 \tag{5.32}$$

$$\forall\, o\ \in O, \forall\, x \in X, m \in H_x$$

$$\sum_{m\,\in H_x} \boldsymbol{x_{omxn}} \leq 1 \tag{5.33}$$

$$\forall\, o\ \in O, \forall\, x \in X, n \in H_x$$

Constraint (5.32) ensures that an ingress wavelength to an optical switch from a given neighbour node of the optical switch is relayed to at most one neighbour node of that optical switch. On the other hand, Constraint (5.33) ensures that an egress wavelength from an optical switch, which is relayed to a neighbour node of the optical switch, entered the switch from only one neighbour node of the optical switch. This avoids wavelength collision at a

given output port of the optical switch. Both constraints (5.32) and (5.33) implement a passive switching matrix for a configurable optical switch.

$$\boldsymbol{d_{omxn}^{ij}} \leq \boldsymbol{v_{omx}^{ij}} \tag{5.34}$$

$$\forall\, o\ \in O, \forall\, i,j\ \in Q, \forall\, x \in X, \forall\, m,n \in H_x$$

$$\boldsymbol{d_{omxn}^{ij}} \leq \mathcal{D}\, \boldsymbol{x_{omxn}} \tag{5.35}$$

$$\forall\, o\ \in O, \forall\, i,j\ \in Q, \forall\, x \in X, \forall\, m,n \in H_x$$

$$\boldsymbol{d_{omxn}^{ij}} \geq \boldsymbol{v_{omx}^{ij}} - \mathcal{D}(1 - \boldsymbol{x_{omxn}}) \tag{5.36}$$

$$\forall\, o\ \in O, \forall\, i,j\ \in Q, \forall\, x \in X, \forall\, m,n \in H_x$$

Constraints (5.34) - (5.36) linearise the derivation of continuous variable $\boldsymbol{d_{omxn}^{ij}}$ which includes a product of a continuous variable and a binary variable as shown in Constraint (5.37).

$$\boldsymbol{d_{omxn}^{ij}} = \boldsymbol{v_{omx}^{ij}}\, \boldsymbol{x_{omxn}} \tag{5.37}$$

$$\forall\, o\ \in O, \forall\, i,j\ \in Q, \forall\, x \in X, \forall\, m,n \in H_x$$

Constraint (5.37) ensures that the traffic that enters an optical switch is relayed according to the path configuration of the optical switch.

Table 5.1: Evaluation of constraint (5.34) - (5.36)

$\boldsymbol{x_{omxn}}$	**(5.34)**	**(5.35)**	**(5.36)**	$\boldsymbol{d_{omxn}^{ij}}$
0	$\boldsymbol{d_{omxn}^{ij}} \leq \boldsymbol{v_{omx}^{ij}}$	$\boldsymbol{d_{omxn}^{ij}} \leq 0$	$\boldsymbol{d_{omxn}^{ij}} \geq \boldsymbol{v_{omx}^{ij}} - \mathcal{D}$	$\boldsymbol{d_{omxn}^{ij}} = 0$
1	$\boldsymbol{d_{omxn}^{ij}} \leq \boldsymbol{v_{omx}^{ij}}$	$\boldsymbol{d_{omxn}^{ij}} \leq \mathcal{D}$	$\boldsymbol{d_{omxn}^{ij}} \geq \boldsymbol{v_{omx}^{ij}}$	$\boldsymbol{d_{omxn}^{ij}} = \boldsymbol{v_{omx}^{ij}}$

Given $\boldsymbol{x_{omxn}}$, Table 5.1 gives the evaluation of constraints (5.34) - (5.36) and the values of $\boldsymbol{d_{omxn}^{ij}}$ that satisfies the defined constraints.

$$\boldsymbol{v_{omx}^{ij}} = \sum_{m \in N : m \in H_x} \boldsymbol{d_{omxn}^{ij}} \tag{5.38}$$

$$\forall\, o\ \in O, \forall\, i,j\ \in Q, \forall\, x\ \in X, \forall\, n \in H_x$$

Constraint (5.38) allows optical switches to route traffic passively over the hybrid inter-rack network. This is achieved by ensuring that the traffic that enters an optical switch from a given neighbour node is relayed to the appropriate output port of the optical switch as specified by the configured routing matrix of the switch.

5.4.3 MILP Model for AOPD-DCN

A similar approach taken to emulate optical switches in the MILP model for EO-NetCoD can be adopted to emulate the wavelength selective switch

(WSS) deployed in AOPD-DCN. Additionally, to simplify model formulation we assume that the ToC-AoD switches in AOPD-DCN are always configured to perform OCS. The inter-cluster AoD switch is setup to perform optical packet switching and forwarding. Since, no limitation is specified for interfaces and bi-directional communication is not considered, constraints (5.21) - (5.24) are not applicable in the intra-rack network of AOPD-DCN. In addition to other network constraints from Section 5.4.1 and 5.4.2 the following constraints are required to represent AOPD-DCN as MILP Model.

$$\sum_{n \in H_m:\, m \neq n} \boldsymbol{q_{omn}} \leq 1 \qquad (5.39)$$

$$\forall\, o \in O, \forall\, m \in A$$

Constraint (5.39) ensures that a given wavelength from a compute node is transmitted only once by that compute node on the inter-rack network.

$$\sum_{n \in H_m:\, m \neq n} \boldsymbol{q_{onm}} \leq 1 \qquad (5.40)$$

$$\forall\, o\ \in O, \forall\, m \in A$$

Constraint (5.40) ensures that a given wavelength received from any neighbour node of a given compute node on the inter-rack network is received only once at the compute node.

$$\sum_{o\, \in O} \boldsymbol{w_{omn}} \leq\ \mathbb{D} \qquad (5.41)$$

$$\forall\, m\ \in A, \forall\, n \in A_m$$

Constraint (5.41) is the capacity constraint of the physical link between two compute nodes in the same rack. Where $\mathbb{D}$ is the maximum transmitting and receiving capacity supported on the point-to-point physical link between nodes on the intra-rack backplane.

5.4.4 Network Setup and Input Parameter

We compare the performance of both variants of NetCoD to AOPD-DCN in the small cluster of a composable DC depicted in Figure 5.6 and Figure 5.7. The cluster comprises of 3 racks, each rack holds 4 compute nodes and a ToR switch. In all evaluation scenario, use of multiple links between two switches to achieve path diversity is not implemented or modelled (as shown in Figure 5.6 and Figure 5.7) for further simplicity. This assumption is practical in this evaluation scenario because a small cluster is considered. A composable DCs with more and bigger clusters would require the

implementation of path diversity between network switches for robustness and to boost network capacity.

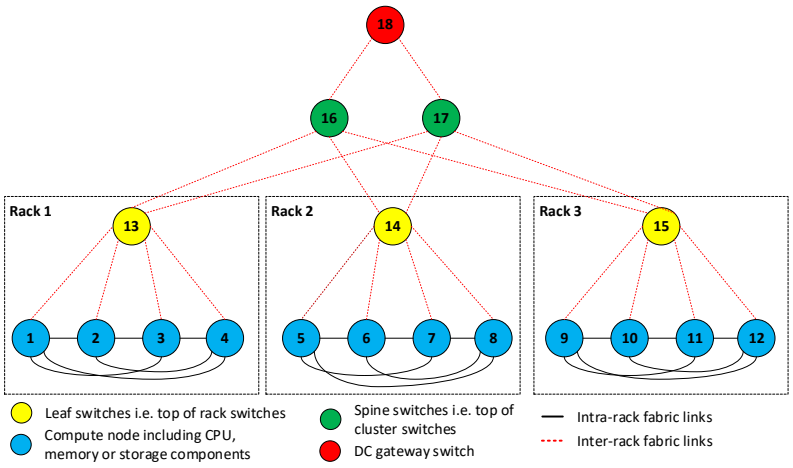


Figure 5.6: A DC cluster with NetCoD

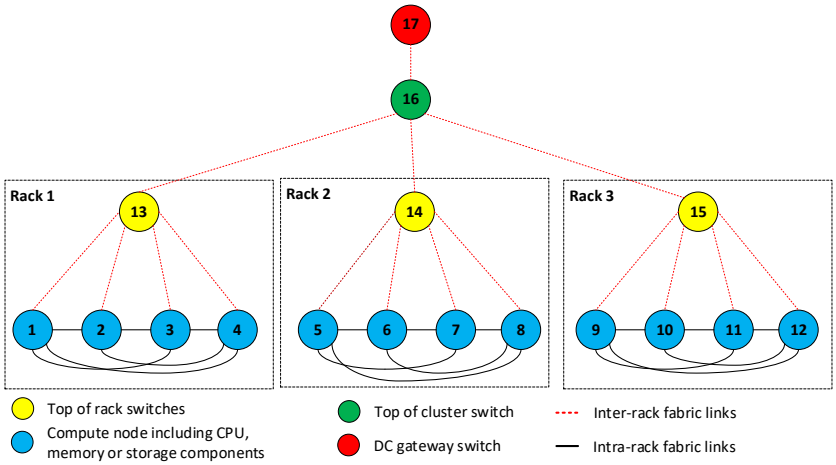


Figure 5.7: A DC cluster with AOPD-DCN

The load proportional energy per bit values of DC network tiers are given in Table 5.2 along with the fixed power consumption of electrical and optical network components. We adopt the energy per bit values predicted for on-board, intra-rack and inter-DC tiers of next generation DC networks as given in [80]. However, because it is expected that the electrical ToR/ToC switch will be relatively more complex than the NCH but less complex than the DC gateway switch. Consequently, the inter-rack energy per bit value suggested in [80] is not completely suitable in our setup. Hence, we conservatively assume that the energy per bit value for each electrical switch in the leaf-spine-layers of the inter-rack network is 5 pJ/b in all evaluation scenarios. This reflects the relative difference in DCN tier complexity. The typical operating (idle) power of all electrical switches in the all network topologies consider is 312 W [103].

The NCH has two functions. Firstly, it has to convert the electrical data streams to an appropriate optical wavelength and transmit this data i.e., the forwarding function. The associated power consumption is typically 1 pJ/b [80]. Secondly, the NCH also has to compute the route to the destination and configure the SOA switches and set up the path. It is assumed here that these operations consume an equal amount of power in the NCH. Spreading this power consumption between all the data streams and wavelengths handled by the NCH leads to a power consumption of 1 pJ/b for path computation and setup. It should be noted that this choice is on the pessimistic side as the path computation tasks can consume much lower power if for example look up tables are used. Similarly, it is also conservatively assumed that path computation and setup functions performed by all electrical switches also leads to a power consumption of 1 pJ/b.

We adopt 100 Gbps for single wavelength transmission in the network topology given recent practical demonstration of such lane rate [107]. We expect even greater single wavelength lane rate for short reach inter-connects in the future as optical technologies advance [108]. To simplify the evaluation scenario, it is assumed that each interface integrated with the NCH of a compute node can emit 4 distinct wavelengths. Hence, a maximum 8 wavelength is supported under both variants of NetCoD. For fair comparison, it is also assumed that the interface used by each compute node in AOPD-DCN to connect to the inter-rack network also supports 8 wavelengths. Hence, enabling 800 Gbps link from each compute node to the ToR switch. Additionally, it is assumed that each interface connected to the intra-rack backplane of AOPD-DCN can transmit or receive four wavelengths in parallel at 100 Gbps lane rate i.e., $\mathbb{D} = 400\,Gbps$. Hence, representing a 400 Gbps transceiver.

The low-energy SOA switch proposed in [109] which has an energy per bit of 15.8 pJ/b at 10 Gbps per single wavelength data rate is adopted to implement the integrated optical switch at each compute node. Since, 100 Gbps single wavelength data rate is adopted, the energy per bit of the SOA switch reduces by a factor of 10 at this data rate. As in AOPD-DCN, low power and configurable WSSs are selected as optical switches in EO-NetCoD. Each WSS has a typical operating power consumption of 50 W [104]. It is important to note that similar network setup and input parameters given in this sub-section are adopted in other sections of this chapter.

Table 5.2: Network input parameters

Parameters	**Value**
On-board network energy per bit	0.1 pJ/b [80]
Intra-rack network energy per bit	1 pJ/b [80]
Electrical switch energy per bit	5 pJ/b
Inter-DC gateway switch energy per bit	10 pJ/b [80]
Energy per bit of routing function.	1 pJ/b
Typical operating (idle) power of electrical switch	312 W [103]
Typical operating power of WSS-based optical switch	50 W [104]
SOA-based switch energy per bit at 100 Gbps single wavelength transmission data rate	1.58 pJ/b
Single wavelength transmission data rate	100 Gbps

5.4.5 Performance Evaluation

5.4.5.1 Maximum Throughput

Under this scenario, Objective 1 of Equation (5.14) is adopted to maximise the throughput of all network topologies being evaluated. The results in Table 5.3 shows that the maximum throughput obtained under bi-directional and unidirectional communication is equal under both variant of NetCoD. All nodes are transmitting at maximum capacity of 800 Gbps. This also confirms that interface constraints are the primary factor that determine the maximum throughput achievable. Hence, to increase total throughput, the number of wavelengths supported by compute nodes' interfaces in both variants of NetCoD must be increased. Adoption of unidirectional communication paths may also be explored to increase network throughput. However, relative to the adoption of unidirectional communication within each rack, results obtained when bi-directional communication is adopted within each rack in the DC is comparable. Hence, given appropriate intelligence in both variants of NetCoD, the size of the intra-rack backplane can be effectively halved via the adoption of bi-directional communication over a single optical link.

AOPD-DCN is also limited by interface constraints since direct point-to-point communication between compute nodes is limited to 400 Gbps. However, compute nodes also utilise the capacity of the inter-rack network to increase overall throughput. Consequently, the maximum throughput of

AOPD-DCN is greater than the throughput of either variant of NetCoD. The maximum transmitting and receiving capacity of the inter-cluster switch of AOPD-DCN is limited. This is because a single fibre connects the ToC switch to the inter-cluster switch in the DC as shown in Figure 5.7. Hence, the maximum transmitting and receiving data rate of the inter-cluster switch is capped at 800 Gbps because only eight (8) unique wavelengths are supported in the network. This limitation is easily remedied by deploying parallel links between the inter-cluster switch and the ToC switch in the AOPD-DCN. The adoption of a leaf-spine topology in the inter-rack of EO-NetCoD mitigates such limitation. Path diversity is an inherent feature of the leaf-spine physical topology. Hence, the maximum transmitting and receiving capacity of DC gateway switch is doubled relative to that of the inter-cluster switch of AOPD-DCN.

Table 5.3: Maximum throughput of network topologies

	AOPD-DCN	**E-NetCoD**		**EO-NetCoD**	
		Bi-directional	**Unidirectional**	**Bi-directional**	**Unidirectional**
Throughput	24.8 Tbps	11.2 Tbps	11.2 Tbps	11.2 Tbps	11.2 Tbps

5.4.5.2 Energy Efficient Network Load Test

We further evaluate and compare the performance of both variants of NetCoD with that of AOPD-DCN by performing energy efficient network load test. The network load test considers the routing and forwarding of input traffic (τ_{sd}) between nodes in the composable DC. A non-uniform traffic distribution is considered between routing and forwarding nodes in the DC. Load between 80 Gbps and 720 Gbps are considered to represent 10% to 90% utilisation of each compute node's maximum throughput when both variants of NetCoD are deployed. The non-uniform traffic distribution (TD) is given by equation (5.42); where Nodes A, B, C and D are compute nodes in the same rack; Node R is a randomly selected compute node in a remote rack to Nodes A-D; and Node G is the gateway switch of the DC.

We consider a scenario where intra-rack, inter-rack and north-south traffics accounts for 80%, 15% and 5% of each node's traffic in either directions respectively as given in equation (5.42). Such traffic distribution pattern is adopted because it is expected that intra-rack traffic will be the dominant traffic within each rack, followed by inter-rack traffic. It is expected that north-south traffic will have the lowest percentage. This is based on the

de facto knowledge that majority of DCN traffic is within the DC while north-south traffic accounts for a small percentage of the total traffic in the DCN.

$$TD = \begin{bmatrix} \boldsymbol{Node} & \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{R} & \boldsymbol{G} \\ \boldsymbol{A} & 0 & 0.7 & 0.1 & 0 & 0.15 & 0.05 \\ \boldsymbol{B} & 0 & 0 & 0.7 & 0.1 & 0.15 & 0.05 \\ \boldsymbol{C} & 0.1 & 0 & 0 & 0.7 & 0.15 & 0.05 \\ \boldsymbol{D} & 0.7 & 0.1 & 0 & 0 & 0.15 & 0.05 \\ \boldsymbol{R} & 0.15 & 0.15 & 0.15 & 0.15 & 0 & 0.05 \\ \boldsymbol{G} & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0 \end{bmatrix} \quad (5.42)$$

An input traffic τ_{sd} to the MILP model is derived by multiplying the load by the traffic distribution as given in Equation (5.43). Relative to the maximum throughput of AOPD-DCN, E-NetCoD and EO-NetCoD obtained in Table 5.3, Figure 5.8 shows network utilisation at all network loads considered.

$$\tau_{sd} = Load \cdot TD \quad (5.43)$$

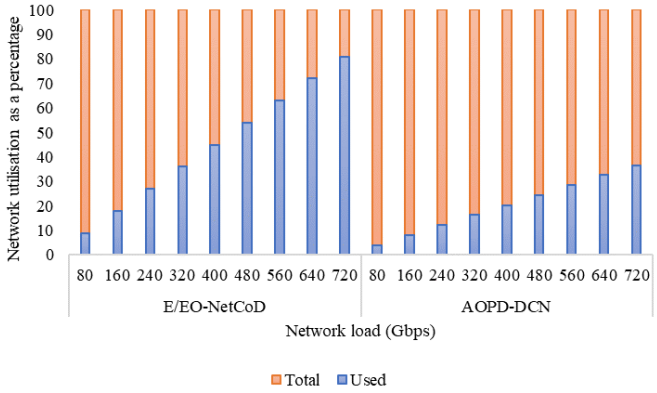


Figure 5.8: Network throughput utilisation under different load

Given the input traffic τ_{sd} the energy efficiency load test is performed by adopting Objective 2 of Equation (5.15). The results in Figure 5.9 show that comparable TNRP is incurred under all network topologies considered. However, marginally higher TNRP is consumed under AOPD-DCN. This is because additional routing intelligence is required to avoid wavelength collision on the link that connects the ToC switch to the gateway switch. Introducing path diversity in the inter-rack network can mitigate such limitations. The inherent path diversity of the leaf-spine physical architecture adopted in the inter-rack network of EO-NetCoD implies that such limitations are mitigated. However, the problem remains a concern. Hence, in a large deployment scenario, adoption of multiple links between switches of the inter-rack network is recommended to further leverage path diversity via spatial multiplexing. It is important to note that the introduction of additional links between switches to achieve diversity in a practical deployment could also enhance load balancing and improve capacity and resilience. Generally, the TNFP consumed by EO-NetCoD is comparable to that of AOPD-DCN as

illustrated in Figure 5.10. The TNFP increases drastically when the E-NetCoD is considered because of the adoption of electrical switches in the network topology.

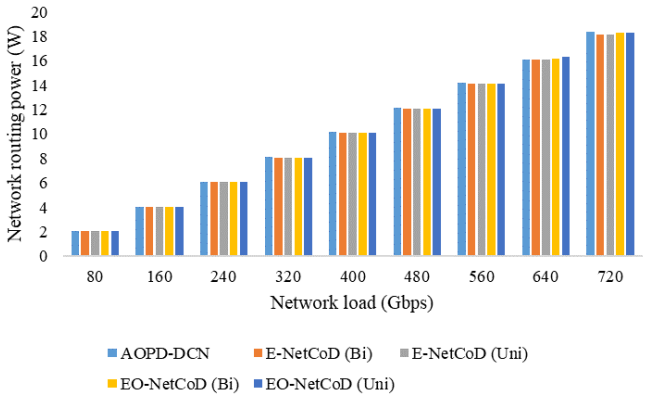


Figure 5.9: Total network routing power

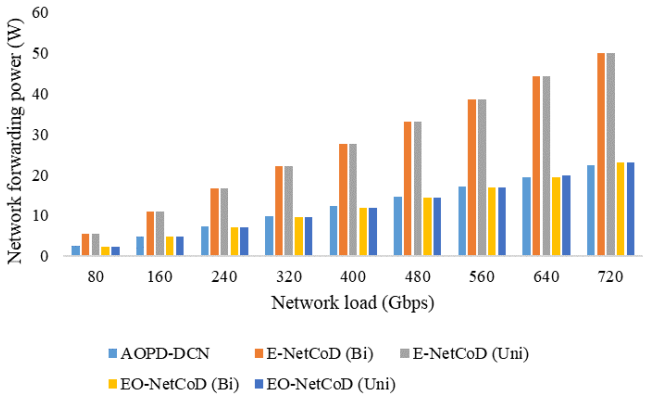


Figure 5.10: Network forwarding power

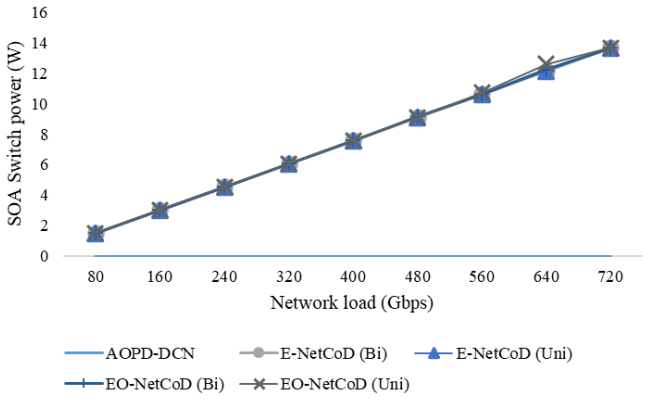


Figure 5.11: SOA switch power

As expected, the power consumed by the SOA switch grows proportionally with the network loads under both variants of NetCoD as illustrated in Figure 5.11. This is because load proportional power profile is adopted for the integrated optical switches. AOPD-DCN does not employ SOA

switches, hence, SOA switch power is zero for the network topology. Furthermore, AOPD-DCN has the lowest total switch operational power (TSWOP) as shown in Figure 5.12. This is because it uses one less optical switch than EO-NetCoD. On the other hand, Figure 5.12 also shows that E-NetCoD has the higher TSWOP because it requires 6 active electrical switches. Each active electrical switch has a corresponding operational power consumption.

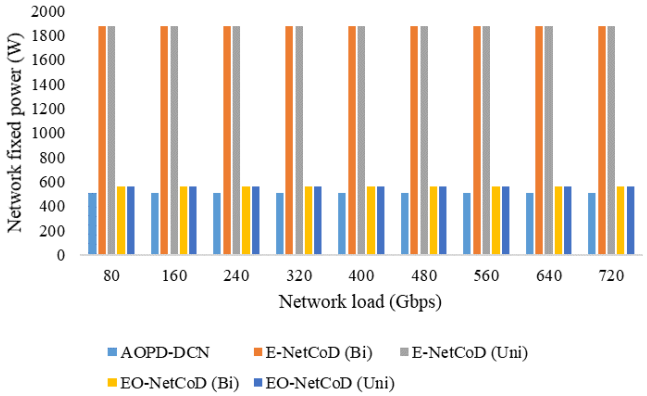


Figure 5.12: Total switch operating power (TSWOP)

5.5 MILP Model for Energy Efficient Placement of VMs

A MILP model that minimises total compute power consumption, TNPC, and VM rejection in a rack-scale composable DC is given in this section. This MILP is an extension of the model given in Section 5.4. The model adds compute related sets, parameters, variables, and constraints to those of the network as given in Section 5.4. Similar to Section 5.4, AOPD-DCN, E-NetCoD and EO-NetCoD network topologies are adopted in the rack-scale composable DC. Furthermore, the MILP model also evaluates the performance of various forms of resource disaggregation in a rack-scale composable DC over different network topologies. Given a specific network topology and the resource demand template of each VM, the model selects the optimum placement for compute resources requested by each VM. The resulting placement ensures the minimisation of total compute power consumption, TNPC and the number of rejected VMs. The MILP model constraints include resource capacity constraints and resource locality constraints in additions to network constraints from Section 5.4. The compute related sets, parameters and variables of the MILP model are given as follows.

Compute Related Sets and Parameters

C Set of CPU resource components

M	Set of memory resource components
S	Set of storage resource components
R	Set of DC racks
$\mathbb{C}_j$	Capacity of CPU component $j \in C$
IC	Idle power as a fraction of maximum CPU power consumption
CP_j	Maximum power consumption of CPU component $j \in C$
ΔC_j	Power factor of CPU component $j \in C$; $\Delta C_j = \frac{CP_j - IC \cdot CP_j}{\mathbb{C}_j}$; in W/GHz
$\mathbb{M}_j$	Capacity of memory component $j \in M$
IM	Idle power as a fraction of maximum memory power consumption
MP_j	Maximum power consumption of memory component $j \in M$
ΔM_j	Power factor of memory component $j \in M$; $\Delta \mathrm{M_j} = \frac{MP_j - IM \cdot MP_j}{\mathbb{M}_j}$; in W/GB
$\mathbb{S}_j$	Capacity of storage component $j \in S$
IS	Idle power as a fraction of maximum storage power
SP_j	Maximum power consumption of storage component $j \in S$
ΔS_j	Power factor of storage component $j \in S$; $\Delta \mathrm{S_j} = \frac{SP_j - IS \cdot SP_j}{\mathbb{S}_j}$; in W/GB
CN_{jn}	$CN_{jn} = 1$ if CPU $j \in C$ is placed in node $n \in N$. Otherwise $CN_{jn} = 0$. Note that CPU components can only be placed in compute nodes.
MN_{jn}	$MN_{jn} = 1$ if RAM $j \in M$ is placed in node $n \in N$. Otherwise $MN_{jn} = 0$. Note that memory components can only be placed in compute nodes
SN_{jn}	$SN_{jn} = 1$ if hard disk drive (HDD) $j \in S$ is placed in node $n \in N$. Otherwise $SN_{jn} = 0$. Note that storage components can only be placed in compute nodes
NR_{nr}	$NR_{nr} = 1$, If node $n \in N$ is placed in rack $r \in R$, otherwise $NR_{nr} = 0$

VM related set and parameters

V	Set of virtual machines
VC_v	CPU demand of VM $v \in V$
VM_v	RAM demand of VM $v \in V$
VS_v	Storage demand of VM $v \in V$
$\mathcal{C}U_v$	CPU to memory (RAM) traffic required by VM $v \in V$
$\mathcal{C}D_v$	Memory (RAM) to CPU traffic required by VM $v \in V$
$\mathcal{S}U_v$	CPU to storage traffic required by VM $v \in V$
$\mathcal{S}D_v$	Storage to CPU traffic required by VM $v \in V$
$\mathcal{I}U_v$	Uplink north-south traffic of VM $v \in V$
$\mathcal{I}D_v$	Downlink north-south traffic of VM $v \in V$
$\mathcal{M}_{sd}$	In-memory computing traffic from VM $s \in V$ to VM $d \in V$
VG_{vn}	$VG_{vn} = 1$ denotes that node $n \in G$ is the gateway node for north-south traffic of VM $v \in V$
α	Cost associated with a VM rejection.

Variables:

$\boldsymbol{c_{vj}}$	$\boldsymbol{c_{vj}} = 1$ indicates that CPU demand of VM $v \in V$ is served by CPU $j \in C$. Otherwise, $\boldsymbol{c_{vj}} = 0$
$\boldsymbol{m_{vj}}$	$\boldsymbol{m_{vj}} = 1$ indicates that RAM demand of VM $v \in V$ is served by memory (RAM) $j \in M$. Otherwise, $\boldsymbol{m_{vj}} = 0$
$\boldsymbol{s_{vj}}$	$\boldsymbol{s_{vj}} = 1$ indicates that storage resource demand of VM $v \in V$ is served by HDD $j \in S$. Otherwise, $\boldsymbol{s_{vj}} = 0$
$\mathbb{c}_j$	$\mathbb{c}_j = 1$ if CPU $j \in C$ is active. Otherwise, $\mathbb{c}_j = 0$
$\mathbb{m}_j$	$\mathbb{m}_j = 1$ if RAM $j \in M$ is active. Otherwise, $\mathbb{m}_j = 0$
$\mathbb{s}_j$	$\mathbb{s}_j = 1$ if HDD $j \in S$ is active. Otherwise, $\mathbb{s}_j = 0$
$\mathbb{a}_n$	$\mathbb{a}_n = 1$ if compute node $n \in A$ is active. Otherwise, $\mathbb{a}_n = 0$
$\boldsymbol{r_r}$	$\boldsymbol{r_r} = 1$ if rack $r \in R$ is active. Otherwise, $\boldsymbol{r_r} = 0$
$\mathbb{r}$	Number of active racks in the composable DC
$\boldsymbol{CM_{vsd}}$	$\boldsymbol{CM_{vsd}} = 1$ if CPU demand of VM $v \in V$ is placed in compute node $s \in A$ and mermory demand of VM $v \in V$ is placed in compute node $d \in A$. Otherwise, $\boldsymbol{CM_{vsd}} = 0$.

$\boldsymbol{CS_{vsd}}$	$\boldsymbol{CS_{vsd}} = 1$ if CPU demand of VM $v \in V$ is placed in compute node $s \in A$ and storage demand of VM $v \in V$ is placed in compute node $d \in A$. Otherwise, $\boldsymbol{CS_{vsd}} = 0$.
$\boldsymbol{z_{sd}^{xy}}$	$\boldsymbol{z_{sd}^{xy}} = 1$ if memory to memory (in-memory computing) traffic exists from VM $x \in V$ in compute node $s \in A$ to VM $y \in V$ in compute node $d \in A$. Otherwise, $\boldsymbol{z_{sd}^{xy}} = 0$.
$\boldsymbol{IR_{sd}}$	Total inter-resource traffic from compute node $s \in A$ to compute node $d \in A$ due to VM demand placement.
$\boldsymbol{EW_{sd}}$	Total east-west traffic from node $s \in N$ to node $d \in N$.
$\boldsymbol{NS_{sd}}$	Total north-south traffic from node $s \in N$ to node $d \in N$.
$\mathbf{T}_{sd}$	Total traffic from node $s \in N$ to node $d \in N$.
$\boldsymbol{\beta_v}$	$\boldsymbol{\beta_v} = 1$ if VM $v \in V$ is rejected. Otherwise, $\boldsymbol{\beta_v} = 0$.
$\boldsymbol{\beta}$	Total number of rejected VMs

Certain variables in the MILP model are derived from other variables. These linear derivations form part of the linear constraints required in the MILP model. Such variables are derived as follows:

$$\boldsymbol{\beta_v} = 1 - \sum_{j \in C} \boldsymbol{c_{vj}} \qquad (5.44)$$

$$\forall\, v \in V$$

Equation (5.44) derives the state of a VM using knowledge of the placement of the workload's CPU demand in any CPU component in the composable DC.

$$\sum_{v \in V} \boldsymbol{c_{vj}} \geq \mathbb{c}_j \qquad (5.45)$$

$$\forall\, j \in C$$

$$\sum_{v \in V} \boldsymbol{c_{vj}} \leq Q\, \mathbb{c}_j \qquad (5.46)$$

$$\forall\, j \in C$$

$$\sum_{v \in V} \boldsymbol{m_{vj}} \geq \mathbb{m}_j \qquad (5.47)$$

$$\forall\, j \in M$$

$$\sum_{v \in V} \boldsymbol{m_{vj}} \leq Q\, \mathbb{m}_j \qquad (5.48)$$

$$\forall\, j \in M$$

$$\sum_{v \in V} \boldsymbol{s_{vj}} \geq \mathbb{s}_{\boldsymbol{j}} \tag{5.49}$$

$$\forall\, j \in S$$

$$\sum_{v \in V} \boldsymbol{s_{vj}} \leq Q\, \mathbb{s}_{\boldsymbol{j}} \tag{5.50}$$

$$\forall\, j \in S$$

Equations (5.45) – (5.50) derive the state of CPU, memory, and storage components. The state of each resource component depends on the utilisation of each resource type to satisfy resource demands of any active VM.

Total CPU Power Consumption

Total CPU power consumption (***TCPC***) in the DC is derived as follows.

$$\boldsymbol{TCPC} = \sum_{j \in C} \left(IC\ CP_j\ \mathbb{c}_{\boldsymbol{j}} + \sum_{v \in V} \Delta C_j\ \boldsymbol{c_{vj}}\ VC_v \right) \tag{5.51}$$

Total Memory Power Consumption

Total memory power consumption (***TMPC***) in the DC is derived as follows.

$$\boldsymbol{TMPC} = \sum_{j \in M} \left(IM\ MP_j\ \mathbb{m}_{\boldsymbol{j}} + \sum_{v \in V} \Delta M_j\ \boldsymbol{m_{vj}}\ VM_v \right) \tag{5.52}$$

Total Storage Power Consumption

Total storage power consumption (***TSPC***) in the DC is derived as follows.

$$\boldsymbol{TSPC} = \sum_{j \in S} \left(IS\ SP_j\ \mathbb{s}_{\boldsymbol{j}} + \sum_{v \in V} \Delta S_j\ \boldsymbol{s_{vj}}\ VS_v \right) \tag{5.53}$$

Total Compute Power Consumption

Total compute power consumption (***TComPC***) in the DC is derived as follows.

$$\boldsymbol{TComPC} = \boldsymbol{TCPC} + \boldsymbol{TMPC} + \boldsymbol{TSPC} \tag{5.54}$$

Derived Network Variables

$$\sum_{v \in V} \sum_{c \in C} \boldsymbol{c_{vc}}\ CN_{jn} + \sum_{v \in V} \sum_{m \in M} \boldsymbol{m_{vm}}\ MN_{mn} + \sum_{v \in V} \sum_{s \in S} \boldsymbol{s_{vs}}\ SN_{sn} \geq \mathbb{a}_{\boldsymbol{n}} \tag{5.55}$$

$$\forall\, n \in A$$

$$\sum_{v \in V} \sum_{c \in C} \boldsymbol{c_{vc}}\ CN_{jn} + \sum_{v \in V} \sum_{m \in M} \boldsymbol{m_{vm}}\ MN_{mn} + \sum_{v \in V} \sum_{s \in S} \boldsymbol{s_{vs}}\ SN_{sn} \leq Q\mathbb{a}_{\boldsymbol{n}} \tag{5.56}$$

$$\forall\, n \in A$$

Equations (5.55) and (5.56) derive the state of each compute node based on the use of CPU, memory, or storage resource in that compute node to satisfy the resource demand of any VM.

$$\sum_{n \in A} \mathbb{a}_{\boldsymbol{n}}\, NR_{nr} \;\geq \boldsymbol{r_r} \tag{5.57}$$

$$\forall\, r \in R$$

$$\sum_{n \in A} \mathbb{a}_{\boldsymbol{n}}\, NR_{nr} \leq Q\, \boldsymbol{r_r} \tag{5.58}$$

$$\forall\, r \in R$$

Equations (5.57) and (5.58) derive the state of each rack based on the state of compute nodes in that rack.

$$\mathbb{r} = \sum_{r \in R} \boldsymbol{r_r} \tag{5.59}$$

Equation (5.59) derives the number of active racks in the DC.

$$\boldsymbol{CM_{vsd}} \leq \sum_{c \in C} \boldsymbol{c_{vc}}\, CN_{cs} \tag{5.60}$$

$$\forall\, v \in V, s \in N, d \in N$$

$$\boldsymbol{CM_{vsd}} \leq \sum_{m \in M} \boldsymbol{m_{vm}}\, MN_{md} \tag{5.61}$$

$$\forall\, v \in V, s \in N, d \in N$$

$$\boldsymbol{CM_{vsd}} \geq \sum_{c \in C} \boldsymbol{c_{vc}}\, CN_{cs} + \sum_{m \in M} \boldsymbol{m_{vm}}\, MN_{md} - 1 \tag{5.62}$$

$$\forall\, v \in V, s \in N, d \in N$$

Equations (5.60) - (5.62) implement the product of two derived binary variables as illustrated in Equation (5.63).

$$\boldsymbol{CM_{vsd}} = \sum_{c \in C} \boldsymbol{c_{vc}}\, CN_{cs} \sum_{m \in M} \boldsymbol{m_{vm}}\, MN_{md} \tag{5.63}$$

$$\forall\, v \in V, s \in N, d \in N$$

Equation (5.63) derives $\boldsymbol{CM_{vsd}}$ which gives the compute nodes where the CPU and memory resource demands of VM $v \in V$ are placed.

$$\boldsymbol{CS_{vsd}} \leq \sum_{c \in C} \boldsymbol{c_{vc}}\, CN_{cs} \tag{5.64}$$

$$\forall\, v \in V, s \in N, d \in N$$

$$\boldsymbol{CS_{vsd}} \leq \sum_{n \in S} \boldsymbol{s_{vn}}\, SN_{nd} \tag{5.65}$$

$$\forall\, v \in V, s \in N, d \in N$$

$$\boldsymbol{CS_{vsd}} \geq \sum_{c \in C} \boldsymbol{c_{vc}}\, CN_{cs} + \sum_{n \in S} \boldsymbol{s_{vn}}\, SN_{nd} - 1 \tag{5.66}$$

$$\forall\, v \in V, s \in N, d \in N$$

Equations (5.64) - (5.66) implement the product of two derived binary variables as illustrated in Equation (5.67).

$$\boldsymbol{CS_{vsd}} = \sum_{c \in C} \boldsymbol{c_{vc}}\, CN_{cs} \sum_{n \in S} \boldsymbol{s_{vn}}\, SN_{nd} \tag{5.67}$$

$$\forall\, v \in V, s \in N, d \in N$$

Equation (5.67) derives $\boldsymbol{CS_{vsd}}$ which gives the compute nodes where the CPU and storage resource demands of VM $v \in V$ are placed.

$$\boldsymbol{z_{sd}^{xy}} \leq \sum_{m \in M} \boldsymbol{m_{xm}}\, MN_{ms} \tag{5.68}$$

$$\forall\, x, y \in V, s \in N, d \in N$$

$$\boldsymbol{z_{sd}^{xy}} \leq \sum_{m \in M} \boldsymbol{m_{ym}}\, MN_{md} \tag{5.69}$$

$$\forall\, x, y \in V, s \in N, d \in N$$

$$\boldsymbol{z_{sd}^{xy}} \geq \sum_{m \in M} \boldsymbol{m_{xm}}\, MN_{ms} + \sum_{m \in M} \boldsymbol{m_{ym}}\, MN_{md} - 1 \tag{5.70}$$

$$\forall\, x, y \in V, s \in N, d \in N$$

Equations (5.68) - (5.70) implement the product of two derived binary variables as illustrated in Equation (5.71).

$$\boldsymbol{z_{sd}^{xy}} = \sum_{m \in M} \boldsymbol{m_{xm}}\, MN_{ms} \sum_{m \in M} \boldsymbol{m_{ym}}\, MN_{md} \tag{5.71}$$

$$\forall\, x, y \in V, s \in N, d \in N$$

Equation (5.71) derives $\boldsymbol{z_{sd}^{xy}}$ which gives a VM pair (x, y) that exchanges in-memory computing traffic via memory-to-memory communication and the corresponding compute nodes $(s\ and\ d)$ where the memory components, which supports memory resource demand of each VM in the pair, are placed.

$$\boldsymbol{IR_{sd}} = \sum_{v \in V} (\boldsymbol{CM_{vsd}}\, \mathcal{C}U_v + \boldsymbol{CM_{vds}}\, \mathcal{C}D_v + \boldsymbol{CS_{vsd}}\, \mathcal{S}U_v + \boldsymbol{CS_{vds}}\, \mathcal{S}D_v) \tag{5.72}$$

$$\forall\, s, d \in A$$

Equation (5.72) derives $\boldsymbol{IR_{sd}}$ which is the inter-resource traffic between node $s \in N$ and node $d \in N$. Resource locality constraints (described later) ensure that nodes s and d are always in the same rack. Furthermore, resource allocation ensures that source and destination nodes (s and d) are compute nodes.

$$\boldsymbol{EW_{sd}} = \sum_{x \in VM} \sum_{y \in VM: x \neq y} \boldsymbol{z_{sd}^{xy}}\, \mathcal{M}_{xy} \tag{5.73}$$

$$\forall\, s, d \in A$$

Equation (5.73) derives $\boldsymbol{EW_{sd}}$ which is the east-west traffic between memory resource components as a result of the placement of memory resource demands of VMs.

$$\boldsymbol{NS_{sd}} = \sum_{v \in V} \sum_{c \in C} (\boldsymbol{c_{vc}}\, CN_{cs}\, \mathcal{JU}_v\, VG_{vd} + \boldsymbol{c_{vc}}\, CN_{cd}\, \mathcal{JD}_v\, VG_{vs}) \tag{5.74}$$

$$\forall\, s, d \in Y$$

Equation (5.74) derives $\boldsymbol{NS_{sd}}$ which is the north-south traffic from node $s \in Y$ to node $d \in Y$ in the DC. Note that the source of south-bound traffic is a gateway switch in the DC. A gateway switch is also the destination of north-bound traffic in the DC.

$$\mathbf{T}_{\boldsymbol{sd}} = \boldsymbol{IR_{sd}} + \boldsymbol{EW_{sd}} + \boldsymbol{NS_{sd}} \tag{5.75}$$

$$\forall\, s, d \in Y$$

Equation (5.75) derives the traffic demand to be routed and forwarded over the composable DC network.

$$\boldsymbol{TOBP} = \sum_{s \in A} \sum_{d \in A: s=d} 2\, \boldsymbol{IR_{sd}}\, OB + \sum_{s \in A} \sum_{d \in Y: s \neq d} \mathbf{T}_{\boldsymbol{sd}}\, OB + \sum_{d \in A} \sum_{s \in Y: s \neq d} \mathbf{T}_{\boldsymbol{sd}}\, OB \tag{5.76}$$

Equation (5.76) derives the total on-board power ($\boldsymbol{TOBP}$) consumption due to the traversal of the on-board fabric by internal, ingress and egress traffic of all compute nodes.

The **TNPC** from Equation (5.13) is revised to include the $\boldsymbol{TOBP}$ as follows in Equation (5.77).

$$\boldsymbol{TNPC} = \boldsymbol{TNFP} + \boldsymbol{TOBP} + \boldsymbol{TNRP} + \boldsymbol{TXNP} \tag{5.77}$$

The model for energy efficient placement of VM in rack-scale composable DC is defined as follows:

Objective 3: Minimise:

$$\boldsymbol{TComPC + TNPC + \alpha \beta} \tag{5.78}$$

Equation (5.78) is the objective of the model for energy efficient placement of VMs in rack-scale composable DC. It minimises the total compute and network power consumption and the number of rejected VMs. α is the cost (measured in Watt) associated each rejected VM. $\alpha \gg 1$ denotes that high cost is associated with each rejected VM.

Subject to:

Compute constraints

$$\sum_{j \in C} \boldsymbol{c_{vj}} \leq 1 \tag{5.79}$$

$$\forall\, v \in V$$

$$\sum_{j \in C} \boldsymbol{c_{vj}} = \sum_{j \in M} \boldsymbol{m_{vj}} \tag{5.80}$$

$$\forall\, v \in V$$

$$\sum_{j \in C} \boldsymbol{c_{vj}} = \sum_{j \in S} \boldsymbol{s_{vj}} \tag{5.81}$$

$$\forall\, v \in V$$

Constraints (5.79) to (5.81) limit the maximum number of nodes that can host CPU, memory, and storage resource demand of a VM to one. This is because neither replication nor slicing of workloads is permitted. The constraints also permit VM rejection in scenarios where resource capacity is limited. The constraints ensure that the VM is fully embedded.

$$\sum_{n \in N} \sum_{j \in C} \boldsymbol{c_{vj}}\, CN_{jn}\, NR_{nr} = \sum_{n \in N} \sum_{j \in M} \boldsymbol{m_{vj}}\, MN_{jn}\, NR_{nr} \tag{5.82}$$

$$\forall\, v \in V, r \in R$$

$$\sum_{n \in N} \sum_{j \in C} \boldsymbol{c_{vj}}\, CN_{jn}\, NR_{nr} = \sum_{n \in N} \sum_{j \in S} \boldsymbol{s_{vj}}\, SN_{jn}\, NR_{nr} \tag{5.83}$$

$$\forall\, v \in V, r \in R$$

Constraints (5.82) and (5.83) are the locality constraints of rack-scale composable DC. They ensure that CPU, memory, and storage components used to provision a given VM are in the same rack but not necessarily in the same compute node.

$$\sum_{v \in V} VC_v \, \boldsymbol{c_{vj}} \leq \mathbb{C}_j \quad \forall \, j \in C \tag{5.84}$$

$$\sum_{v \in V} VM_v \, \boldsymbol{m_{vj}} \leq \mathbb{M}_j \quad \forall \, j \in M \tag{5.85}$$

$$\sum_{v \in V} VS_v \, \boldsymbol{s_{vj}} \leq \mathbb{S}_j \quad \forall \, j \in S \tag{5.86}$$

Constraints (5.84) - (5.86) denote resource capacity constraints for each CPU, memory, and storage components. These equations also ensure that a compute resource's capacity is conserved.

5.6 Energy Efficient Placement of VMs in Single Rack Setup

Using the combined MILP models from Section 5.4 and Section 5.5, energy efficient placement of VMs in rack-scale composable DCs is studied. Rack-scale DCs that implement logical, hybrid and physical resource disaggregation are considered. A zero-cost-and-un-capacitated network and non-zero-cost-and-capacitated networks (i.e., AOPD-DCN, E-NetCoD or EO-NetCoD) are deployed. Two classes of CPU, memory, and storage resource components (illustrated in Table 5.4) are considered to reflect heterogeneity of compute resources deployed in production DCs. To study the performance of all network topologies within a rack, a composable DC with a single rack is considered in this section. Allocated to the single rack are 8 CPU, 8 memory and 8 storage components. These components are distributed into compute nodes according to the resource disaggregation approach adopted as shown in Figure 5.13.

There are eight heterogenous compute nodes in the single rack of a logically disaggregated composable DC. Each heterogeneous compute node comprises of one CPU, one memory and one storage resource components as shown in Figure 5.13a. The single rack comprises of 10 compute nodes when hybrid disaggregation is adopted in the DC as shown in Figure 5.13b. Compute nodes 1 – 4 are heterogeneous nodes, each comprising of one CPU, one memory and one storage resource components. Compute nodes 5 – 10 are homogenous, each homogenous node comprise of two CPU or two memory or two storage resource components of the same component class. The single rack in a physically disaggregated DC comprises of 12

homogenous compute nodes. Each compute node comprises of two CPU or two memory or two storage resource components from the same class as shown in Figure 5.13c. Because the MILP model's complexity grows as the number network nodes increases, we further simplify the MILP model in single rack scenario. This is achieved by excluding spine switches or ToC switches from corresponding network topologies. A scenario where the ToR switch is connected directly to the DC gateway switch is considered. Furthermore, the network parameters from Section 5.4.4 are adopted in this section.

Table 5.4: Compute component capacity and peak power

Component Type	**CPU**		**Memory**		**Storage**	
Component ID	**Capacity**	**Peak power**	**Capacity**	**Peak power**	**Capacity**	**Peak power**
1	3.6 GHz	130 W	32 GB	40 W	320 GB	6.19 W
2	2.66 GHz	95 W	24 GB	30.72 W	250 GB	6.19 W
3	3.6 GHz	130 W	32 GB	40 W	320 GB	6.19 W
4	2.66 GHz	95 W	24 GB	30.72 W	250 GB	6.19 W
5	3.6 GHz	130 W	32 GB	40 W	320 GB	6.19 W
6	2.66 GHz	95 W	24 GB	30.72 W	250 GB	6.19 W
7	3.6 GHz	130 W	32 GB	40 W	320 GB	6.19 W
8	2.66 GHz	95 W	24 GB	30.72 W	250 GB	6.19 W

Since, the complexity of the model also grows with the number of VMs and size of the composable DC, only few VMs are considered for placement in the single rack. We consider 12 input VMs with a mix of compute resource demand intensity as illustrated in Table 5.5. Furthermore, the data rate of each VM's CPU-memory communication, CPU-disk communication and north-south as also given in Table 5.5. Traffic demands of VMs in Table 5.5 are generated via uniform distribution of total CPU-to-memory traffic, CPU-to-storage traffic and CPU-to-IO traffic over specific ranges. VMs are clustered into in-memory communication groups (IMCG), as illustrated in Table 5.5, to represent sets of related VMs in conventional DCs. Each set of related VMs have one-to-one, one-to-many, many-to-many, or mixed in-memory computing traffic patterns between the applications of the group. The range of

in-memory computing traffic between two VMs in the same group is 5 Gbps to 40 Gbps.

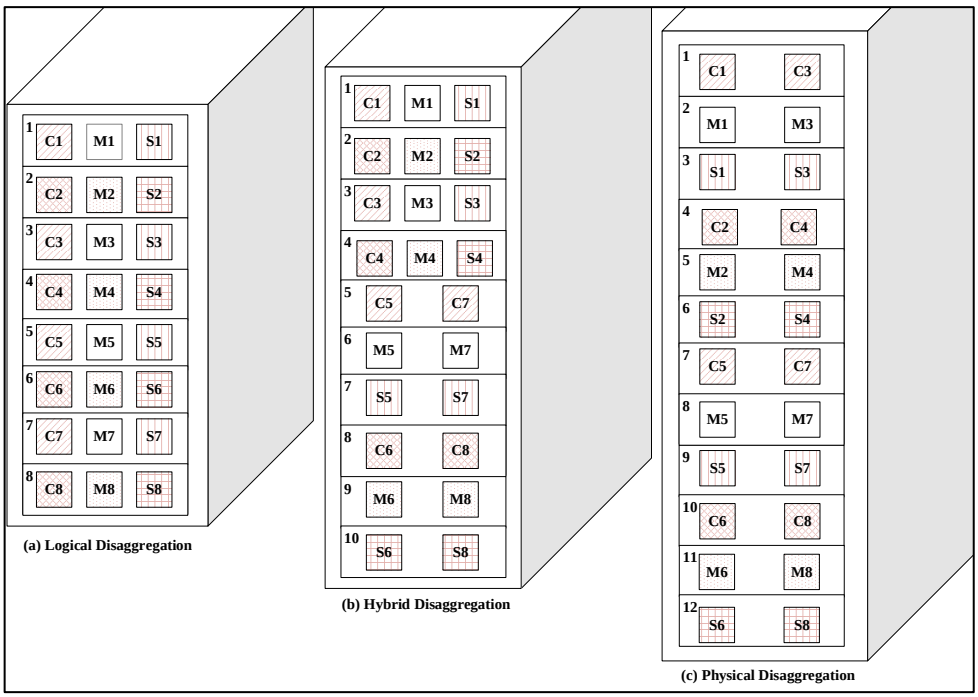


Figure 5.13: Resource disaggregation of single DC rack.

Table 5.5: VM compute and network demands

VM (IMCG)	CPU demand (GHz)	Memory demand (GB)	Storage demand (GB)	CPU – RAM (Gbps)		CPU – HDD (Gbps)		CPU – IO (Gbps)	
				Egress	Ingress	Egress	Ingress	Egress	Ingress
1 (A)	1.8	7.2	80	116.7	50	26	9.3	9.5	6.6
2 (A)	1.8	24	240	50	66.7	60	9.5	3.3	4
3 (A)	2.6	10.8	120	100	41.7	64	6	5	3.5
4 (A)	0.9	13	160	266.7	116.7	86	5	3	4
5 (A)	0.9	3.6	160	466.7	100	23	9	10	2
6 (B)	2.6	32	160	466.7	50	20	28	2.6	2.5
7 (B)	1.8	24	80	333.3	44.4	64	19.5	2.75	3.5
8 (B)	2.6	10.8	80	133.3	233.3	17.5	14	1.7	5.7
9 (B)	2.6	32	80	166.7	66.7	10	29	1	8.5
10 (B)	1.8	7.2	160	116.7	100	14	49	2	3
11 (C)	1.8	24	240	433.3	66.7	68	45	1.75	3
12 (C)	2.6	10.8	80	333.3	25	22	9.7	4	2

The MILP model is solved using the 64-bit AMPL/CPLEX solver on the ARC3 supercomputing node with 24 CPU cores and 128 GB of memory [70]. Our analysis of results from the model focuses on metrics such as total computing power consumption, total network power consumption, number of active resource components, average active resource component utilisation and network available throughput utilisation. To obtain optimal results, the MILP model bin-packs VMs resource demands into compute components to achieve optimal power and utilisation efficiencies within compute capacity constraints and network constraints.

5.6.1 Zero-Cost-and-Un-capacitated Network

Under this scenario, energy efficient placement of VM is performed in rack-scale DCs that employ logical, physical or hybrid disaggregation over an un-capacitated-and-zero-cost network. The zero-cost-and-un-capacitated network has no network capacity constraints and zero power is consumed as result of traffic forwarding and routing over network. Results in Figure 5.14, Figure 5.15 and Figure 5.16 show that all disaggregation approaches achieved optimal efficiencies. The compute resource components are utilised based on available capacity and energy efficiency. VM resource demands are bin-packed into CPU, memory, and storage components to avoid VM rejection. Bin-packing also achieved optimal energy efficiency under the corresponding form of resource disaggregation employed in the rack.

Since any form of network cost is absent under this scenario, optimal compute energy efficiency is achieved under all forms of disaggregation employed in the single rack. Hence, the utilisation of active resource components is maximised within the component's available capacity and their corresponding energy efficiency. Figure 5.15 shows that equal number of CPU, memory and storage components are activated under all forms of resource disaggregation employed in the DC. While all (8) CPU and memory components in the rack are activated to prevent VM rejection, only 6 storage components are activated. This is because the storage resource demands of VM as given in Table 5.5 are less intensive relative to the capacity of storage components considered. Hence, to promote greater energy efficiency, consolidation of storage resource demands into 320 GB storage components is preferred. Fewer 250 GB storage components are activated as shown in Figure 5.14. This is because the 320 GB storage component is more energy efficient than the 250 GB storage component. The 320 GB storage can support higher capacity at the same peak power consumption as the 250 GB storage component.

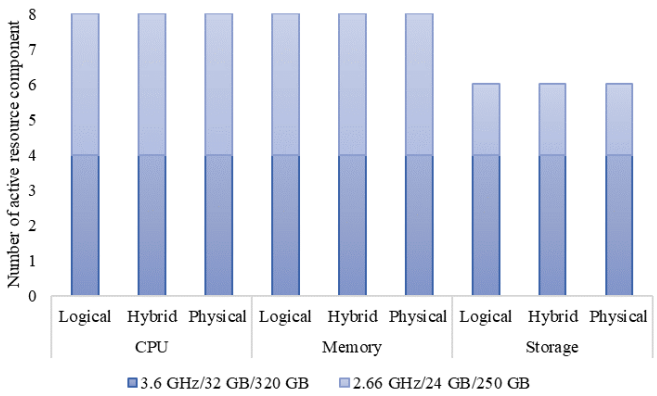


Figure 5.14: Number of active DC compute components

Furthermore, equal average active resource component utilisation is obtained when all forms of resource disaggregation are employed in the rack-scale DC as shown in Figure 5.15. Equal TCPC, TMPC and TSPC are also obtained under all disaggregation types considered as shown in Figure 5.16. This is a consequent of optimal utilisation of CPU, memory and storage resource components under all forms of resource disaggregation employed in the rack. The fact that the same results are obtain under all resource disaggregation approaches does not always imply that the placement of VM resource demands is the same under different forms of disaggregation. Hence, as is the case in our scenarios, different VM placements may achieve the desired optimal performance provided that resource component capacity constraint is satisfied. The results obtained under this scenario demonstrates the efficacy of all forms of disaggregation to achieve optimal efficiency when an arbitrary un-capacitated-and-zero-cost network is available. However, it is expected that the results will change when network constraints and cost are present in the composable DCs.

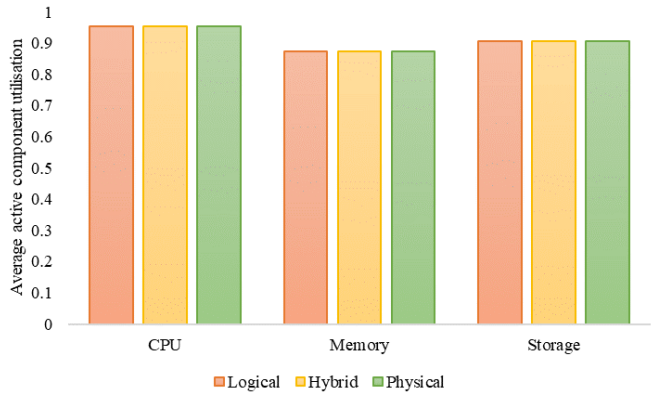


Figure 5.15: Average utilisation of active DC compute components

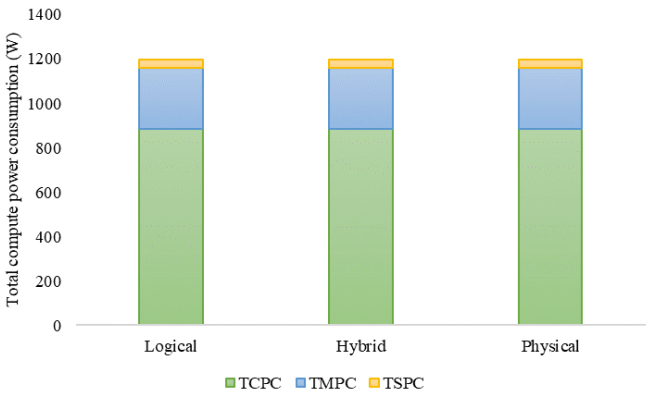


Figure 5.16: Total compute power consumption

5.6.2 Non-Zero-Cost-and-Capacitated Networks

A non-zero-cost-and-capacitated network is a network with non-zero-power consumption and network capacity constraints. As defined in the objective function of the MILP model, there are three primary factors that influence placement of VM resource demands in non-zero-cost networks. These are VM rejection, compute energy efficiency and TNPC. However, because a high cost is associated with the rejection of all VMs in the DC, rejection of VMs is strongly discouraged. Hence, to achieve optimal results when a non-zero-cost-and-capacitated network is employed in the rack-scale DC, best-effort will be made to prevent VM rejection while trade-offs between compute energy efficiency and network energy efficiency are also considered. Placement of VMs resource demands is also expected to be constrained by both compute and network constraints stated in the MILP model.

5.6.2.1 Logical Disaggregation

When logical disaggregation is considered in the rack-scale composable DC, similar VM placement is replicated for all network topologies considered as illustrated in Table 5.6. Hence, optimal placement of VM resource demands does not change with the network topology that is adopted. However, relative to the zero-cost-and-un-capacitated network, the placement of VMs when AOPD-DCN, E-NetCoD or EO-NetCoD is deployed is sub-optimal. This is because of the presence of network constraints and the introduction of network power consumption. Even though, the resulting traffic matrix generated under un-capacitated-and-zero-cost network can be routed through AOPD-DCN, it was not preferred. This is because the additional network power that must be consumed to achieve the same compute energy efficiency as the zero-cost network scenario outweighs the potential benefits that could be achieved. Therefore, an alternative VM placement strategy is adopted when AOPD-DCN is deployed to minimise TNPC. On the other hand, the

resulting traffic generated under the zero-cost-and-un-capacitated network scenario is unrouteable by both variants of NetCoD. Because of the interface capacity constraint at compute nodes. Hence, an alternative VM placement strategy is required to satisfy network constraints and to minimise TNPC when both variants of NetCoD are deployed.

Table 5.6: VM placement in logically disaggregated rack

Scenarios	**Zero-Cost**			**AOPD-DCN**			**E-NetCoD**			**EO-NetCoD**		
VM (IMCG)	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**
1 (A)	3(3)	5(5)	7(7)	1(1)	8(8)	1(1)	7(7)	2(2)	7(7)	1(1)	4(4)	1(1)
2 (A)	7(7)	3(3)	1(1)	1(1)	4(4)	1(1)	7(7)	6(6)	7(7)	1(1)	6(6)	1(1)
3 (A)	4(4)	2(2)	8(8)	7(7)	7(7)	7(7)	1(1)	1(1)	1(1)	5(5)	5(5)	5(5)
4 (A)	3(3)	4(4)	2(2)	5(5)	7(7)	5(5)	3(3)	1(1)	3(3)	7(7)	5(5)	7(7)
5 (A)	5(5)	8(8)	3(3)	7(7)	7(7)	7(7)	1(1)	1(1)	1(1)	5(5)	5(5)	5(5)
6 (B)	5(5)	1(1)	5(5)	5(5)	5(5)	5(5)	3(3)	3(3)	3(3)	7(7)	7(7)	7(7)
7 (B)	1(1)	6(6)	1(1)	3(3)	3(3)	3(3)	5(5)	5(5)	5(5)	3(3)	3(3)	3(3)
8 (B)	8(8)	8(8)	2(2)	2(2)	2(2)	4(4)	4(4)	4(4)	5(5)	8(8)	8(8)	3(3)
9 (B)	6(6)	7(7)	3(3)	4(4)	1(1)	4(4)	6(6)	7(7)	6(6)	6(6)	1(1)	6(6)
10 (B)	7(7)	3(3)	5(5)	3(3)	3(3)	3(3)	5(5)	5(5)	5(5)	3(3)	3(3)	3(3)
11 (C)	1(1)	5(5)	7(7)	6(6)	6(6)	6(6)	8(8)	8(8)	8(8)	2(2)	2(2)	2(2)
12 (C)	2(2)	4(4)	3(3)	8(8)	8(8)	3(3)	2(2)	2(2)	6(6)	4(4)	4(4)s	6(6)
# of active components	8	8	6	8	8	6	8	8	6	8	8	6
C – CPU Component ID, M – Memory Component ID, S – Storage Component ID, N – Compute Node ID												

A strategy that balances the trade-offs between compute power consumption and TNPC is adopted to obtain optimal placement. Energy efficient placement of CPU demands is often given higher priority. Hence, CPU demands are consolidated (within resource capacity constraint as much as possible) to achieved high utilisation of active CPU components. However, to ensure that TNPC is minimised, memory demands of some VMs are placed in the same compute node as the CPU demand. This strategy is strongly applied to VMs that are known to have very high total CPU-to-memory traffic as shown in Table 5.6. It is also applied to some VMs that are known to have moderately high CPU-to-memory traffic, provided that the capacity of the memory component in the corresponding compute node is enough. Otherwise, the memory demand of such VMs is placed in different node to

achieve lower TMPC by ensuring high utilisation of active memory components. The memory demand of VMs that have low-medium volume of total CPU-to-memory traffic are often provisioned in a different way. They are usually provisioned in memory components that are in a different compute node from the CPU component that host the CPU demand. This strategy is adopted to ensure high utilisation of active memory components; consequently, TMPC is also minimised. The adopted strategy also reduces the impact of resource disaggregation on the network by minimising the TNPC because lower volume of traffic is sent over the network.

Additionally, TNPC is further minimised by placing the storage demand of most VMs in the same compute node as the CPU demand of the VM as illustrated in Table 5.6. This reduces the volume of traffic in the network and the consequential power that would have been consumed. However, the storage demand of some VMs are also placed in a different compute node from the node hosting the VM's CPU demand to reduce the TSPC by optimally utilising active storage components. This also reduces the number of active storage components as illustrated in Figure 5.17.

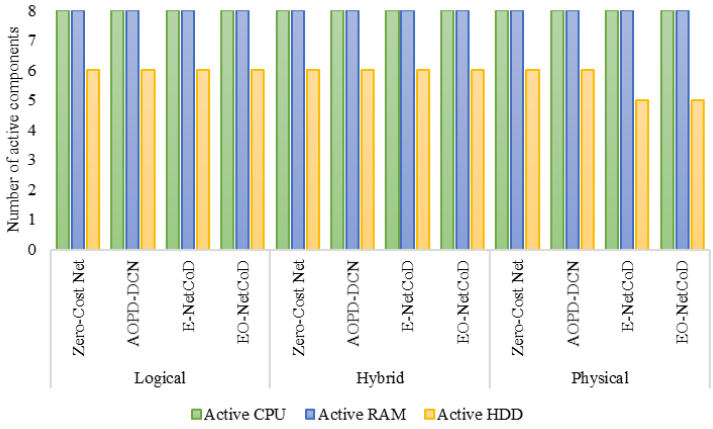


Figure 5.17: Number for active resource component in DC

It is also important to note that, network traffic is also minimised by reducing or eliminating in-memory communication between VMs in the same IMCG. This is done by placing such VMs into the same memory component. For instance, the memory demand of VMs 3, 4, and 5 which belong to the IMCG-A are always placed in the same memory component as illustrated in Table 5.6. As a result, in-memory communication between such VMs is avoided. This is because the CPU allocated to each VM can be granted access to the appropriate address on the common (shared) memory component to retrieve desired data. However, because in-memory communication volume is relatively small, the reduction of the volume of in-

memory communication in the network does not have a strong effect on VM demand placement decisions.

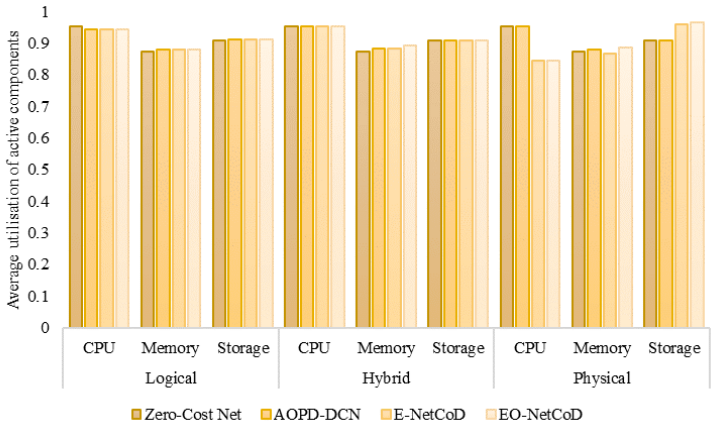


Figure 5.18: Average utilisation of active components

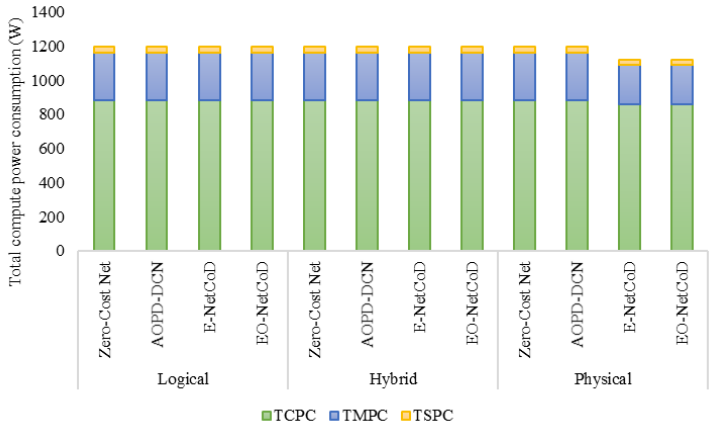


Figure 5.19: Total compute power consumption

Relative to the zero-cost-and-un-capacitated network, the total compute (CPU, memory, and storage) power consumption increased marginally when non-zero-cost-and-capacitated networks are deployed in a logically disaggregated rack. Marginal increase in TCPC, TMPC and TSPC are responsible for the marginal increase in total compute power consumption. The less efficient placement of VM demands when AOPD-DCN, E-NetCoD or EO-NerCoD is deployed is responsible for the fall in average utilisation of active CPU components as shown in Figure 5.18. The fall in the average utilisation of active CPU components is somewhat marginal. Compared to the zero-cost-and-un-capacitated network scenario, the average utilisation of active memory and storage components marginally increases as shown in Figure 5.18. The marginal increase in average utilisation of active memory and storage components does not result in a fall in TMPC and TSPC respectively. This is because it is achieved by increasing the utilisation of

memory and storage components that are less energy efficient. Hence, the TMPC and TSPC increased marginally in-spite of the increase in average active memory and storage utilisation.

The results obtained under the logical disaggregation setup provides an excellent basis for another fair comparison of the various network topologies being considered. The TNRP is the same for AOPD-DCN and both variants of NetCoD. This is because the traffic demand between two communicating nodes pair is always routed directly between the two nodes in the logical layer of all network topologies considered. Hence, all end-to-end logical paths created over the physical topologies (AOPD-DCN, E-NetCoD and EO-NetCoD) are direct and do not employ intermediated routing nodes along the path. In the physical layer, the TNFP consumed by AOPD-DCN and EO-NetCoD are equal. This is because the optical ToR switch is the only intermediate node traversed by end-to-end light paths created over the physical topology. Since, the optical switch only has a constant (low) operating power consumption that is non-load proportional, it has no impact on the TNFP of both AOPD-DCN and EO-NetCoD. On the other hand, the E-NetCoD topology employs an electrical switch which has both fixed operational and load proportional components in its power profile. Hence, relative to the AOPD-DCN or EO-NetCoD topologies, the E-NetCoD topologies has higher TNFP. The ToR switch is an important intermediate node traversed by some direct logical layer links created between communicating nodes pairs.

The TOBP is the same across all three topologies because the VMs are placed in the same way under all topologies. SOAPC makes no contribution to the TNPC of AOPD-DCN since the SOA switches are not required at each compute node. On the other, the same SOAPC is obtained under both variants of NetCoD. Hence, the contribution of SOAPC to the TNPC is the same as shown in Figure 5.20. The TSWOP of AOPD-DCN and EO-NetCoD are equal as both topologies employed one optical switch and one electrical gateway switch with equal typical operating power consumption. The TSWOP of the E-NetCoD is higher as illustrated in Figure 5.20 because two electrical switches are required when the topology in deployed in this evaluation scenario.

The optimal VM placement obtained via MILP optimisation enabled zero-hop communication between all intra-rack communicating node pairs. This achieved a balanced trade-off between compute power consumption and TNPC under all network topologies considered. Furthermore, single hop communication is employed for communication between compute nodes and

the gateway switch. Hence, the SOAPC and/or the TNRP and TNFP consumed due to VMs placement are minimised under the appropriate topology. The TNPC increases by 0.7% when AOPD-DCN is replaced by EO-NetCoD. The energy efficient SOA switches employed in EO-NetCoD are solely responsible for the marginal increase in TNPC observed. The TNPC increased by 71% when EO-NetCoD is replaced by E-NetCoD. The TSWOP and the TNFP consumed by the electrical ToR switch, which is an important intermediate node, are responsible for this relative increase in TNPC as shown in Figure 5.20.

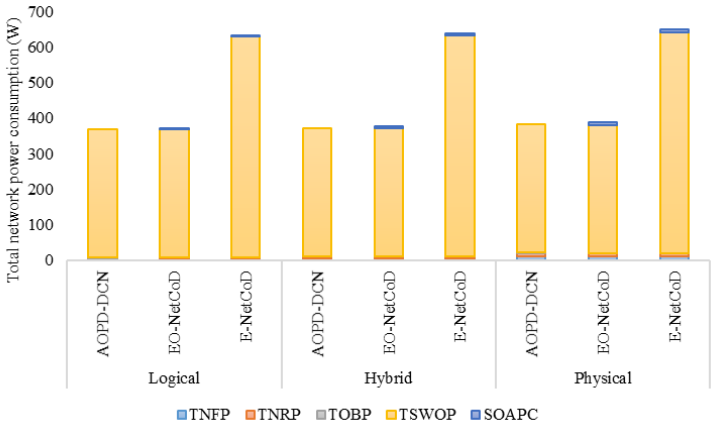


Figure 5.20: Total network power consumption in DC

However, evaluation of power consumption alone is not sufficient to evaluate network performance. The AOPD-DCN topology adopts a generic design approach to achieve a suitable network for composable DC. Hence, in a logically disaggregated DC that employs AOPD-DCN, each compute node requires multiple dedicated interfaces to ensure full mesh physical connectivity in the rack. This becomes a design problem as the number of nodes in each rack increases. This is because multiple (up to 48 high-capacity) interfaces must be fitted onto each compute node. Both variant of NetCoD mitigate this problem by adopting a targeted design approach that addresses the specific challenge posed by resource disaggregation in a practical composable DC.

Relative to AOPD-DCN, the maximum throughput achievable by both variants of NetCoD is significantly lower as shown in Figure 5.21. However, it is important to remember that in AOPD-DCN, each compute node has a dedicated 400 Gbps interface to communicate directly with each co-rack compute node. In addition, each compute node in AOPD-DCN also has a total of 800Gbps that is available to communicate via the inter-rack network. On the other hand, each compute node has a maximum of 800 Gbps to

communicate with all nodes in the DC when either variant of NetCoD is considered. However, both variants of NetCoD achieve significantly higher utilisation of the available network throughput as shown in Figure 5.22. About 20% of E-NetCoD and EO-NetCoD available capacity is used when logical disaggregation is implemented in the composable rack as illustrated in Figure 5.22. On the other hand, higher throughput provided by AOPD-DCN significantly exceeds the practical need in the logically disaggregated DCs. Only about 5% of AOPD-DCN available throughput is used as illustrated in Figure 5.22. Therefore, relative to the technically challenging generic design approach adopted for AOPD-DCN, the specific and simpler design approach adopted for NetCoD achieves greater utilisation (4 times greater) while delivering similar performance.

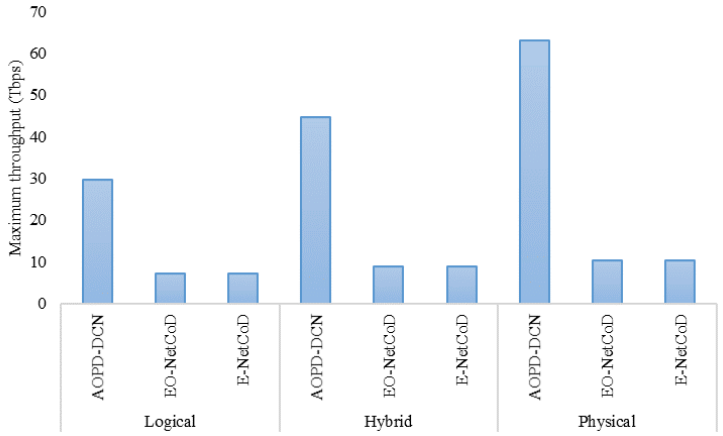


Figure 5.21: Maximum network throughput in single rack composable DC

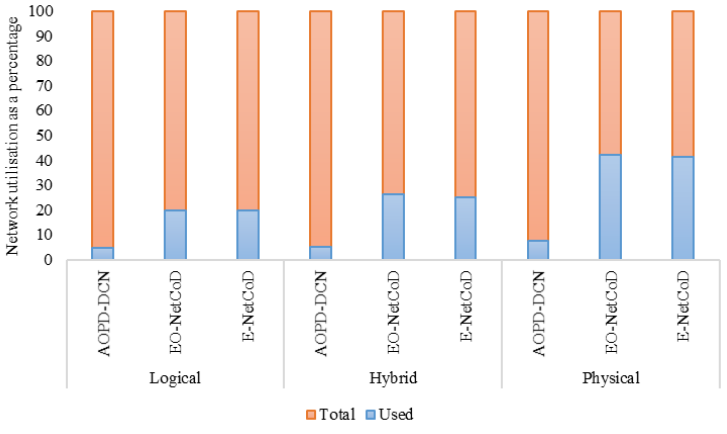


Figure 5.22: Network throughput utilisation in single rack composable DC

5.6.2.2 Hybrid Disaggregation

The results show that the general placement strategy observed under in the logically disaggregated rack-scale DC is implemented when hybrid disaggregation is employed in the rack-scale composable DC. That is, VM

rejection remains discouraged as a balance between compute energy efficiency and network energy efficiency is found. However, relative to the zero-cost-and-un-capacitated network, total compute power consumption is marginally (below a percent) higher when AOPD-DCN, E-NetCoD or EO-NetCoD is employed. This is because maximum compute energy efficiency is not achieved when network cost and constraints are introduced. Given both compute and network constraints, marginal concessions in compute energy efficient are required to achieve an optimal result.

Relative to the zero-cost-and-un-capacitated network, the total compute power consumption obtained when AOPD-DCN is deployed in rack-scale composable DC that implements hybrid disaggregation is marginally (less that a percent) higher. A marginal increase in the TMPC is solely responsible for the marginal increase in total compute power consumption. Revisions in the placement of VMs' memory resource demands as a result of network constraints is responsible for the increase in TMPC. Although the same number of memory resource components are utilised, and the average active utilisation of memory component increased as shown in Figure 5.17, a less energy efficient memory component is highly utilised. Therefore, the TMPC increased accordingly as shown in Figure 5.19. Relative to the zero-cost-and-un-capacitated network, the placement of VMs' CPU and storage demands is different, as shown in Table 5.7, when AOPD-DCN was deployed the rack-scale DC. However, Figure 5.19, Figure 5.17, and Figure 5.18 respectively show that the revised placement achieved the same TCPC and TSPC as zero-cost-and-un-capacitated network. Equal number of active CPU and storage resources and the same average active resource utilisation are also achieved.

The placement of VMs obtained when E-NetCoD is deployed is a replica of the placement obtained when AOPD-DCN was deployed in the rack-scale composable DC. Hence, equal number of active resource components is obtained under both scenarios and the average utilisation of active resource CPU, memory and storage components is equal under both scenarios. Consequently, equal total compute power consumption (TCPC, TMPC and TSPC) is obtained when E-NetCoD is deployed to replace AODP-DCN in a rack-scale DC that implements hybrid disaggregation. However, the resulting placement of CPU demands obtained under E-NetCoD is different from the placement of CPU demands obtained under the zero-cost-and-un-capacitated network as illustrated in Table 5.7.

Table 5.7: VMs placement in a rack that implements hybrid disaggregation

Scenarios	Zero-Cost			AOPD-DCN			E-NetCoD			EO-NetCoD		
VM (IMCG)	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**
1 (A)	5(5)	5(6)	5(7)	3(3)	3(3)	3(3)	3(3)	3(3)	3(3)	5(5)	4(4)	5(7)
2 (A)	5(5)	4(4)	5(7)	7(5)	8(9)	5(7)	5(5)	8(9)	7(7)	5(5)	8(9)	5(5)
3 (A)	3(3)	2(2)	8(10)	8(8)	2(2)	2(2)	6(8)	2(2)	4(4)	8(8)	2(2)	4(4)
4 (A)	1(1)	8(9)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)
5 (A)	3(3)	6(9)	1(1)	1(1)	1(1)	7(7)	1(1)	1(1)	5(7)	1(1)	1(1)	5(7)
6 (B)	4(4)	3(3)	3(3)	5(5)	7(6)	7(7)	8(8)	7(6)	5(7)	7(5)	7(6)	5(7)
7 (B)	7(5)	1(1)	4(4)	7(5)	6(9)	5(7)	5(5)	6(9)	2(2)	3(3)	6(9)	3(3)
8 (B)	8(6)	8(9)	8(10)	2(2)	2(2)	2(2)	4(4)	4(4)	4(4)	2(2)	2(2)	2(2)
9 (B)	2(2)	7(6)	7(7)	6(8)	5(6)	4(4)	7(5)	5(6)	7(7)	6(8)	5(6)	2(2)
10 (B)	7(5)	1(1)	3(3)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)
11 (C)	1(1)	5(6)	7(7)	3(3)	3(3)	3(3)	3(3)	3(3)	3(3)	3(3)	3(3)	3(3)
12 (C)	6(8)	6(9)	4(4)	4(4)	4(4)	4(4)	2(2)	2(2)	2(2)	4(4)	4(4)	4(4)
# of active components	8	8	6	8	8	6	8	8	6	8	8	6
C – CPU Component ID, M – Memory Component ID, S – Storage Component ID, N – Compute Node ID												

The placement of VMs when EO-NetCoD implemented is comparable to the placement of VMs obtained when E-NetCoD is deployed in the composable DC. Equal TCPC and TSPC are obtained under both scenarios. This is because equal number of CPU and storage components are activated under both scenarios. The average utilisation of active CPU and storage component obtained when EO-NetCoD is deployed is equal to the corresponding values obtained when E-NetCoD is implemented in the DC. 8 RAM components are active under both scenarios. However, a slight variation in the placement of memory demands under EO-NetCoD is responsible for marginal increase in the TMPC relative to the corresponding value obtained under E-NetCoD. The average utilisation of active memory component under EO-NetCoD is marginally higher as shown in Figure 5.18. This is because a memory component with less energy efficiency is highly utilised compared to when E-NetCoD is deployed in the composable DC.

The results obtained by solving the MILP model also show attempts to minimise TNPC when AOPD-DCN, E-NetCoD and EO-NetCoD topologies are employed in the DC. AOPD-DCN primarily employs zero-hop communication between intra-rack compute nodes to ensure that both TNFP and TNRP are

minimised. However, in one instance where the capacity of the direct interface between nodes is limited, an additional path is established via the optical ToR switch to support intra-rack traffic. Note that it is more energy efficient to setup a light-path via the optical switch than to use another compute node as an intermediate node. This is because of the associated routing and forwarding power that will be consumed at the intermediate compute node. In the physical layer, direct light-paths are setup between each compute node and the gateway switch via the optical ToR switch. The direct light-paths facilitate the transmission of north-south traffic in both northbound and southbound directions.

As observed when AOPD-DCN is employed in the DC, direct light-paths are often established between communicating node pairs in the DC to carry traffic when EO-NetCOD is employed. However, multi-hop communication is also periodically used to transmit low data rate (mice) traffic to ensure optimal utilisation of provisioned light-paths. Such low data rate (mice) traffic is piggybacked on other light-paths that are established to convey low-medium data rate traffic to an intermediate node. The intermediate node thereafter sets-up another lightpath to jointly forward the transiting mice traffic and its own traffic to a final destination node. Multi-hop communication paths can be provisioned in both the logical and physical layers of the corresponding network topology. It is important to note that a large traffic demand between two nodes maybe divided into mice and elephant portions to ensure optimal utilisation of the network. On the one hand, the mice portion maybe forwarded via multi-hop communication path to optimally utilise the network by maximising the utilisation of each active lightpath. On the other hand, the elephant portion of the divided traffic is forwarded via zero-hop communication path to reduce the SOAPC, TNRP and TNFP. This is because forwarding elephant traffic via multi-hop communication paths significantly increases TNPC.

E-NetCoD adopts a similar approach as EO-NetCoD to maximise utilisation the active light-paths and to mitigate the impact of network constraints. When the E-NetCoD is employed in rack-scale composable DC, direct virtual links are setup between source and destination nodes of the traffic demand in the logical layer. However, in the physical layer, mice flow of the virtual layer is often forwarded via multi-hop communication path. Such flows are carried on a common lightpath with low-medium sized flows that are destined for the selected intermediate compute node. At the intermediate compute node, the transiting mice flow is piggybacked on a different lightpath

that is setup to convey another low-medium sized flow originating at the intermediate compute node. This strategy is commonly observed for mice flows that originate from compute nodes that comprise of multiple CPU components. Such nodes are highly constrained when the CPU demands of multiple workloads are placed in them. Hence, multiple light-paths must be provisioned to convey inter-resource traffic to and from storage and memory components within the rack. Light-paths must also be provisioned to support north-south traffic to and from the inter-rack network in the DC. On the other hand, the elephant flows are usually forwarded via zero-hop communication path to reduce the SOAPC, TNRP and TNFP.

Generally, in the DC that implements hybrid disaggregation, AOPD-DCN has the lowest TNPC, as seen in Figure 5.20. This is because multi-hop communication is reduced and AOPD-DCN does not require SOA switches. However, AOPD-DCN has inherent technical implementation challenges that must be addressed in a practical scenario. Even if such challenges are addressed, an implementation of AOPD-DCN will be grossly underutilised as shown in the Figure 5.22. In a rack-scale DC that implements hybrid disaggregation, the utilisation of the available throughput under either variant of NetCoD is greater than similar results obtained under AOPD-DCN. Relatively, the utilisation of available throughput is 5 times greater under both variant of NetCoD as shown in Figure 5.22.

EO-NetCoD has lower TNPC compared E-NetCoD, as seen in Figure 5.20. This is because energy intensive electrical ToR switches (with relatively high load proportional PC and operational PC) are not used in EO-NetCoD. TOBP is the same under all topologies considered because similar placement strategy adopted. The strategy ensures that maximum traffic is exchanged via the highly energy efficient onboard fabric under all topologies. This strategy helps to minimise TNPC. As expected, Figure 5.20 shows that the TNFP of E-NetCoD is the highest because an electrical ToR is used. The TNFP of EO-NetCoD is higher than that of AOPD-DCN as shown in Figure 5.20. This is because single-hop communication (via other compute nodes) is infrequently adopted to optimally utilise network capacity under EO-NetCoD. This leads to additional forwarding and routing power that are absent when AOPD-DCN is employed. Furthermore, both variants of NetCoD require SOA switches, which also introduce additional network power while AOPD-DCN does not.

For all network topologies considered, relative to when the DC was logically disaggregated, the TNPC is marginally higher in a DC that implements hybrid disaggregation as seen in Figure 5.20. Relative to the

logically disaggregated DC, the TNPC of a DC that implements hybrid disaggregation increased by 0.9%, 0.7% and 1.3% when AOPD-DCN, E-NetCoD and EO-NetCoD are employed respectively. This is because the traffic in higher tiers of the network increases when hybrid disaggregation is employed. Consequently, both forwarding and routing power increase accordingly as shown in Figure 5.20. Furthermore, more SOA switches are required in both variants of NetCoD when hybrid disaggregation is implemented in the composable DC. TNPC only increases marginally when hybrid disaggregation is implemented instead of logical disaggregation. This is because next generation energy efficiency values are adopted for different tiers of the network. Hence, the transmission of significantly higher volumes of traffic when hybrid disaggregation is implemented in the composable DC does not lead to drastic increase in TNPC. It is important to note that comparable compute power consumption is achieved when logical or hybrid disaggregation is adopted in a rack-scale composable DC as shown in Figure 5.19.

5.6.2.3 Physical Disaggregation

In a physically disaggregated DC, the introduction of network constraints and cost changes in the placement of VM demands relative to VM placement obtained under zero-cost-and-un-capacitated network. A review of the placement of VM demands in Table 5.8 shows this. The results obtained when AOPD-DCN is deployed show that the placement strategy gives higher priority to energy efficient utilisation of active CPU components. This strategy is adopted to achieve optimal energy efficiency in the DC. This is because CPU components consume more power than other components in the DC. The placement of CPU demands obtained when AOPD-DCN is deployed in a physically disaggregated DC is different from similar results obtained under the zero-cost-and-un-capacitated network. However, the same TCPC is achieved under both scenarios as shown in Figure 5.19. Similarly, equal number of active CPU component and average active CPU component utilisation are obtained under both scenarios as shown in Figure 5.17 and Figure 5.18 respectively.

Table 5.8 also shows that the memory demand of VMs, which belong to a common IMCG, are placed in the same compute node when AOPD-DCN is implemented. Hence, physical disaggregation is leveraged to reduce and/or eliminate in-memory communication traffic in the network. This kind of placement is more feasible under the physical disaggregation scenario. Consequently, this also minimises forwarding power in the network and can

also reduce multi-hop communication. In most situation, memory demands are placed in a manner that ensures that VMs in the same IMCG are placed in the same compute node. This minimises in-memory communication in the composable DC. Relative to the zero-cost-and-un-capacitated network, the placement strategy adopted for memory demands is responsible for a marginal (less than 1%) rise in the TMPC under AOPD-DCN. This further highlights the need for marginal concessions in compute energy efficiency to achieve optimal overall efficiency

Table 5.8: VMs placement in physically disaggregated rack

Scenarios	**Zero-Cost**			**AOPD-DCN**			**E-NetCoD**			**EO-NetCoD**		
VM (IMCG)	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**	**C(N)**	**M(N)**	**S(N)**
1 (A)	1(1)	5(8)	5(9)	7(7)	1(2)	1(3)	1(1)	8(11)	1(3)	7(7)	6(11)	1(3)
2 (A)	3(1)	7(9)	7(9)	5(7)	1(2)	5(9)	7(7)	6(11)	3(3)	7(7)	8(11)	7(9)
3 (A)	7(7)	6(11)	6(12)	8(10)	3(3)	8(12)	8(10)	8(11)	2(6)	5(7)	5(8)	3(3)
4 (A)	5(7)	8(11)	1(3)	5(7)	3(3)	2(6)	3(1)	3(2)	5(9)	5(7)	6(11)	5(9)
5 (A)	5(7)	8(11)	5(9)	1(1)	3(3)	3(3)	7(7)	8(11)	1(3)	1(1)	5(8)	3(3)
6 (B)	4(4)	1(2)	1(3)	1(1)	5(8)	3(3)	-	-	-	-	-	-
7 (B)	5(7)	5(8)	4(6)	3(1)	2(5)	7(9)	2(4)	5(8)	3(3)	4(4)	2(5)	5(9)
8 (B)	2(4)	4(5)	5(9)	2(4)	4(5)	8(12)	3(1)	3(2)	1(3)	1(1)	5(8)	1(3)
9 (B)	6(10)	3(2)	3(3)	6(10)	7(8)	5(9)	5(7)	1(2)	7(9)	8(10)	3(2)	5(9)
10 (B)	1(1)	7(8)	4(6)	3(1)	8(11)	7(9)	1(1)	5(8)	5(9)	3(1)	1(2)	1(3)
11 (C)	3(1)	2(5)	3(3)	7(7)	6(11)	1(3)	6(10)	4(5)	7(9)	6(10)	4(5)	4(6)
12 (C)	8(10)	6(11)	7(9)	4(4)	8(11)	7(9)	4(4)	2(5)	2(6)	2(4)	1(2)	7(9)
# of active components	8	8	6	8	8	6	8	8	5	8	8	5
C – CPU Component ID, M – Memory Component ID, S – Storage Component ID, N – Compute Node ID												

A different storage demand placement under the AOPD-DCN achieved the same efficiency as the zero-cost-and-un-capacitated network. This is because most storage demands are non-intensive as illustrated in Table 5.5. Hence, bin-packing storage demands for maximum energy efficiency is highly feasible while achieving optimal total efficiency. Consequently, only necessary, and minimal amount of storage components are activated when AOPD-DCN is employed in the physically disaggregated DC. The same number and type of storage component are activated under both AOPD-DCN and the zero-cost-and-un-capacitated network as shown in Figure 5.17. The

average active storage component utilisation is also equal under both scenarios as shown Figure 5.18. Consequently, the same TSPC is obtained under both scenarios as seen in Figure 5.19.

When feasible, attempts are made to maximise network utilisation under AOPD-DCN via coordinated placement of CPU, memory, and storage demands. Such coordination ensures that active light-paths are shared to improve their utilisation. In the logical layer direct virtual links is setup between all communicating nodes pairs. Intra-rack traffic is often sent via direct point-to-point links between compute nodes in the rack. However, additional light-paths may be provisioned over the inter-rack network to supplement the point-to-point capacity in the intra-rack network. This often occurs when the traffic demand between two nodes in the same rack exceeds the capacity of the direct link (i.e., 400 Gbps) that connects two compute nodes together. This strategy is common for compute nodes which hold multiple CPU components. This is because the direct 400 Gbps link between compute nodes may be inadequate for very large or aggregated CPU-to-memory traffic in a composable DC in such scenarios. A direct virtual link (optical light path) is created between the gateway switch and each compute node that has north-south traffic in either northbound or southbound direction. The optical ToR switch serves as a transparent intermediate node between compute nodes and the gateway switch in the physical layer. It is also important to note that some provisioned direct lightpath are poorly utilised when AOPD-DCN is employed. However, since, there is no penalty for poorly utilised lightpath under AOPD-DCN this is an acceptable outcome.

The placement of VMs as illustrated in Table 5.8 also shows that a VM is rejected when both variants of NetCoD are deployed in the physically disaggregated DC. Limited network capacity at compute nodes is responsible for such VM rejection. However, rejection is easily mitigated via the introduction of additional compute node in the rack. Moreover, compared to the small evaluation scenarios considered in this chapter, typical DCs usually have over-provisioned hardware capacity. This mitigates this kind of rejection and ensures that service level agreements at met. Hence, in practice such rejection is unlikely to occur. Although results under both variants of NetCoD also show that it is important to give high priority to energy efficiency of CPU resource component. However, given limited number of CPU compute nodes considered, satisfying the CPU demand of all VMs under either variant of NetCoD while enforcing network constraints is not feasible. Hence, VM rejection is unavoidable. Furthermore, since a common cost of rejection is

associate with all VMs considered, the illustration in Table 5.8 shows that VM 6, is rejected under E-NetCoD and EO-NetCoD. As seen in Table 5.5, VM 6 has very high compute and network requirement

Compared to results obtained under E-NetCoD, the TMPC and TSPC obtained under EO-NetCoD are marginally higher. On the other hand, equal TCPC is obtained under both variants of NetCoD. Equal number of active CPU, memory and storage components were obtained under both scenarios as illustrated in Figure 5.17. The average utilisation of each component is also comparable under both scenarios as illustrated in Figure 5.18. Generally, the TCPC, TMPC and TSPC obtained under both variants of NetCoD are lower compared to similar values obtained under zero-cost-and-un-capacitated network and AOPD-DCN as shown in Figure 5.19. VM rejection under both variants of NetCoD is responsible for this trend.

The results also demonstrate the importance of making strategic placement of resource demands to achieve optimal result while satisfying network constraints. A common strategy employed under both variants of NetCoD is to systematically place resource demands into the rack in a manner that ensures the satisfaction of network constraints. Resource demand placement also attempts to minimise the number of communicating nodes pair in the rack. However, this can be especially difficult for compute nodes with multiple CPU resource in a physically disaggregated rack-scale composable DC. Such compute nodes can host CPU demand of multiple VMs. They also must support the aggregated CPU-to-memory, CPU-to-storage, and CPU-to-gateway traffic in both directions for all VMs placed in them. Multi-hop communication is employed to mitigate stringent network constraints by ensuring that data traffic is optimally aggregated on provisioned light-paths. Multi-hop communication is adopted in two instances as observed when hybrid disaggregation was implemented in the composable DC.

In the first instance, mice traffic (such as CPU-to-storage and CPU-to-gateway traffic) originating from compute node are often forwarded via multi-hop communication path. Such mice flows are aggregated and sent over a lightpath established to convey low-medium data rate traffic to intermediate compute node. The intermediate compute node receives and processes the traffic that is destined for it. It subsequently forwards transiting traffic to the next hop on the multi-hop communication path.

In another instance, large-sized and/or aggregated CPU-to-memory traffic from a compute node can be divided into multiple streams to be forwarded on optical light-paths. The single wavelength data rate (100 Gbps)

is used as the divisor. On the one hand, larger (elephant) portions of such traffic are transmitted via zero-hop paths to minimise network power consumption by maximising active light path utilisation. On the other hand, small (mice) portion of such traffic are forwarded over multi-hop paths to maximise network utilisation and also to mitigate the impact of stringent network constraints. However, it is important to note that adoption of multi-hop communication in this manner can lead to performance degradation since CPU-to-memory traffic type is known to be latency sensitive.

The results obtained when E-NetCoD is employed in the physically disaggregated DC show that direct virtual links are created between all communicating node pairs. Hence, traffic is not relayed in the virtual layer of the network. In the physical layer, high data rate (elephant) flows of each virtual link are sent via direct physical links. Smaller (mice) flows of each virtual link are piggybacked on established light-paths between compute nodes. This helps to promote greater lightpath utilisation. However, a practical implementation must ensure that latency sensitive traffic, such as CPU-memory communication, are sent over minimal number of hops. Generally, southbound traffic from the gateway switch to compute nodes is transmitted over direct light-paths. Such light-paths are established from the gateway switch (via the optical ToR) to each compute node that will receive such traffic. Results show that the same routing and forwarding strategies implemented when E-NetCoD is deployed in physically disaggregated DC are repeated when EO-NetCoD is deployed in a similar DC.

In comparison with a DC that implements logical or hybrid disaggregation, TNPC is marginally higher in a physically disaggregation DC as seen Figure 5.20. Compared to the power consumption of a logically disaggregated DC that employed AOPD-DCN, E-NetCoD and EO-NetCoD, the TNPC increased by 3.7%, 2.6% and 4.4% respectively when the same DC is physically disaggregated. Similar comparison between a DC that implements hybrid disaggregation and physical disaggregation shows that the TNPC of AOPD-DCN, E-NetCoD and EO-NetCoD increased by 2.8%, 1.8% and 3.1% respectively when a DC is physically disaggregated. Increase in TNRP and TNFP due to increase in network traffic traversing the intra-rack network is primarily responsible for the observed results under all network topologies considered. Increase in the SOAPC when physical disaggregation is deployed in the rack also contributes to the increase in TNPC when either variant of NetCoD is used.

Compared to both variants of NetCoD, the TNFP and TNRP obtained when AOPD-DCN is deployed in a rack that implements physical disaggregation is higher. This is because all VMs are provisioned when AOPD-DCN was deployed. On the other hand, a VM is rejected when both variants of NetCoD were deploy in the DC. A similar reason justifies lower TOBP under both variants of NetCoD relative to AOPD-DCN. Figure 5.22 shows the utilisation of the available network throughput of both variants of NetCoD is over 5 times greater than that of AOPD-DCN when a physically disaggregated rack-scale DC is considered. Hence, it is expected that the cost of implementing AOPD-DCN will outweigh the practical benefits that are derived. On the other hand, both variants of NetCoD can provide the required network capacity to support physical disaggregation using a more specific and practical design. This is in contrast to the general-purpose design adopted for AOPD-DCN. However, great intelligence is required to achieve optimal efficiency.

5.7 Summary

In this chapter, the description of two variants of a novel network for composable DCs (NetCoD) was given. The proposed topology leveraged optical technologies and techniques to reduce complexity and cost of a suitable network for composable DCs. In contrast to a general-purposed design employed by a reference network, NetCoD adopts a more targeted design. Using a MILP model for capacitated networks, we compare the performance of both variants of NetCoD to that of a reference network topology in a DC with multiple racks. The electrical-optical variant of NetCoD achieved comparable network energy efficiency as the reference topology. But the energy efficiency of the electrical variant of NetCoD is relatively lower.

Subsequently, we extended the MILP model formulation to consider energy efficient placement of VMs in rack-scale composable DCs. The composable DCs implemented logical, hybrid and physical disaggregation. Relative to the reference topology, the results showed that both variants of NetCoD achieved similar compute energy efficiency under all types of disaggregation considered. The range of scenarios considered highlighted various strategies that can be deployed in practical implementations of either variant of NetCoD. The strategies improved overall energy efficiency in composable DCs while satisfying both compute and network constraints. Under all network topologies considered, a logically disaggregated DC achieved the best results. Across all network topologies evaluated in this

chapter, the average increase in TNPC is 1% when hybrid disaggregation is implemented instead of logical disaggregation in the small evaluation scenario considered. 3.6% is the average increase in TNPC observed when physical disaggregation is implemented instead of logical disaggregation. Additionally, the utilisation of available network throughput by both variants of NetCoD exceeds that of reference topology. This value was 4 – 5 times greater under the different types of disaggregation considered in the rack-scale composable DC.

Chapter 6 : Disaggregation for Energy Efficient Fog in Future 6G Networks

6.1 Introduction

The disaggregation concept that promises greater energy and utilisation efficiencies in large composable DCs. It may also promote greater efficiencies in present and future edge/fog computing infrastructures. In this chapter, we conduct a study to evaluate the impact of adopting resource disaggregation in the fog computing layer of the cloud-of-things architecture. We consider fog network performance influencing factors such as network delay and fog application delay requirements and end-user distribution. We develop an extensive MILP model for the study that compares the adoption of disaggregated servers in the fog computing tier to the adoption of traditional servers. Finally, we develop a fast and scalable heuristic for practical deployment in large scenarios. The heuristic mimics and verifies the MILP model.

6.2 Fog Networks and Related Works

6.2.1 Fog Networks

The fog computing layer is an intelligent intermediate layer between centralised cloud computing servers and geo-distributed connected devices and end-users. This layer provides distributed computing infrastructure at network edges (i.e., metro and access networks) to support connected things and end-users. The fog computing layer complements the centralised hyper-scale cloud computing infrastructure. It extends cloud-like services closer to end-users and connected things for improved performance and to support new classes of applications. The fog computing layer reduces application response time, the volume of communication network traffic and the workload on public cloud infrastructure. Consequently, the fog computing layer can also enable reductions in computing and network infrastructures [110]–[112]. The fog computing paradigm enhances the performance of some existing and emerging applications such as IoT, content delivery network, artificial intelligence, and data analytics. Furthermore, the fog computing layer enables a suitable environment for future internet applications and services.

In recent times, notable wired communication equipment vendors have included extra computation capacity in network routers/switches to support the hosting of non-network related functions and application. For example,

Cisco's catalyst 9000 series switches support application hosting capabilities to host fog applications at the edge of the network [113]. In the wireless communication domain, the concept of mobile edge computing (i.e., a use case of fog computing) is expected to feature in the 5G mobile network infrastructure which is in the early deployment phase. Furthermore, it is also predicted that this concept of mobile edge computing will become more prominent in future 6G networks. Machine Learning (ML) and Artificial intelligence (AI) are predicted to be key features of 6G network infrastructure [114]. Hence, edge computation is required to provide seamless access to ML/AI capabilities at the network edge in the 6G era.

Traditionally, connected devices and end-users in the "things tier" are at the farthest edge of the network. Hence, they interact directly with quasi-distributed cloud computing tier via communication networks (i.e., access, metro and core networks). The introduction of the fog computing tier expands the traditional architecture of the cloud of things continuum [8], [10] by one tier. The traditional definition of fog computing classifies any device with compute, storage and network connectivity as a fog node [11]. However, some locations and devices maybe more optimal than others because of factors such as energy efficiency, resource capacity, node availability, resource reusability and utilisation efficiency. For example, the things tier, which generally comprises of sensors and actuators, is often characterised by small installed computation capacity and limited network connectivity. Hence, resource availability and utilisation efficiency at the things tier is often limited. Addressing these limitations by increasing device form-factor and expanding network connectivity in the things tier might be an overkill for the purpose. On the other hand, increasing the number of end-devices to scale computation capacity at the things tier will further increase the total cost of ownership (TCO). This will also increase the power consumption and carbon footprint associated with the tier. Hence, reducing computation capacity at the extreme edge of the network to its optimal limits is important for sustainable growth in the 6G fog networking and computing era. However, application specific requirements must inform such reductions.

The placement of fog computational capacity in traditional edge network nodes could also improve sustainability. For examples, central offices (COs) and radio cell sites (CSs) could be consider. These nodes a of nodes can be easily accessed by multiple connected devices and end-users. Such strategy also provides effective support for expected high mobility of end-users and connected devices in the fog networking and computing era [12],

[115], [116]. The federation of traditionally independent (fog) computing nodes at the edge of the network can improve efficient usage such computing capacity. This can be achieved via coordinated orchestration and control of the distributed fog nodes over the network infrastructure. Such a group of linked fog nodes with non-application specific resources is called a federated fog network [118]. A federated fog network must also be designed to satisfy application specific requirement. For example, fog applications can have different delay requirement. Some mission critical fog applications such as vehicle-to-everything and industrial process control may require sub-milliseconds (end-to-end) delay, and at most a delay of 20ms or less to be practical. On the other hand, other fog applications may be moderately sensitive to delay [118].

The rising demand for the federation of distributed fog nodes (i.e., fog networks) is motivating the emergence of service-oriented access models for computing resource in the fog computing era. This could be in the form of Fog Infrastructure as a Service (FIaaS) or Fog as a Service (FaaS) [118], [119]. Fog networks can also enable new revenue sources which can offset the TCO incurred by providers that deploy fog computing nodes [118]. The ecosystem of federated distributed fog nodes can also benefit from the adoption of the emerging concept of server disaggregation. This can enable greater efficiency as the concept matures. This chapter studies the impact of using disaggregated servers in a metro-access network of distributed fog nodes.

6.2.2 Related Works

The authors of [110]–[112] conducted extensive studies on how the fog computing layer can enable significant power savings in the cloud of things architecture relative to the adoption of a 2 tier architecture. The work in [10] also showed that a 3 tier cloud of things architecture is better than a 2 tier cloud of things architecture especially when energy consumption and latency are used as comparison metrics. Hence, a 3-tier cloud of things architecture is adopted in this chapter. However, the goal is to minimise the number of computational nodes in the lowest tier of the 3-tier architecture by employing the concept of server disaggregation for the first time in this context.

In [8], the authors gave a description of the fog computing architecture. The benefits and limitations of using the fog layer as a middleware between the cloud and numerous IoT devices were highlighted. Performance evaluation showed that the execution of large tasks experiences significant processing delay in the fog computing layer. Hence, scaling fog computing capacity at the expense of higher financial cost is required. In [118], a platform

which orchestrates distributed fog nodes over the network to form fog networks was proposed. The fog network supported on-demand deployment of applications and services. The authors in [120] introduced a programming model for present and emerging geo-distributed, massive and latency-sensitive applications. The PaaS programming model called “Mobile Fog” provides a simplified programming abstraction and supports on-demand scaling of applications at runtime to use resources in cloud or fog computing tiers.

In [11], the authors proposed a high-level policy for placing applications in the fog computing era based on application latency requirements only. The policy is generic and does not consider the impact of factors such as energy efficiency, resource utilisation, networks, and disaggregation on optimal application placement. Workload offloading and workload assignment are two different approaches used to study the minimisation of response time in fog computing era [12]. Given a set of fog workloads, the workload assignment approach attempts to assign such workloads to fog computing nodes while optimising a specific cost such as response time or energy. On the other hand, the workload offloading approach aims to design policies that offload fog computing requests to other fog nodes or to the cloud while optimising a specific cost. In [12], [121], the authors explored the workload offloading approach by proposing a general delay minimising policy for fog nodes to offload IoT application requests to other fog nodes or to forward these requests to the cloud. Using an analytical model, the authors evaluated the policy and showed that the proposed policy reduces response time for IoT applications. The treatment in [115] adopted the workload assignment approach by using a mathematical model formulation. The formulation compared fog computing and traditional cloud computing in the IoT era using criteria such as power consumption, cost and latency. Results from the model showed that a cloud of things architecture with fog computing as middleware outperforms an architecture without fog computing layer only when there are many latency-sensitive applications. Otherwise, it is better not to deploy the fog computing layer to ensure that overhead cost, with little or no performance benefit, is not incurred. In [122], the authors studied the trade-off between power consumption and latency in the fog-cloud computing system by formulating a workload placement problem i.e., using the workload assignment approach. Simulation results from the formulated problem showed that fog computing can significantly improve the performance of cloud computing by reducing communication latency.

The workload assignment approach is adopted in this chapter by formulating a MILP model to assign varying classes of interactive fog applications. This is because the workload offloading approach may be less appropriate for interactive workloads with stringent delay requirements. A heuristic is proposed from the insights obtained from the MILP model. In contrast to existing literatures, this work is focused on the fog computing tier only and its associated access and metro networks. The things and cloud computing layers of the cloud of thing architecture are not explicitly considered. However, the impact of cloud destined network traffic on the overall performance of the fog computing system is modelled. A cost is also associated with fog servers that are deployed very close to the things layer. Like the works of the authors in [110]–[112], [115], this work explores power consumption, cost and latency in the fog computing layer. However, the novelty of this work is that it also considers disaggregation of fog servers and considers different classes of delay sensitive (interactive) applications. The gains that can be achieved over traditional practice if fog computing nodes adopt disaggregated servers (DSs) over traditional servers (TSs) are evaluated. Other factors that may influence performance in such a setup are also studied.

DC networks is not considered in this chapter to simplify the evaluation scenario. Moreover, Chapters 4 and 5 have given good attention to DC networks for energy efficient composable computing infrastructures. Hence, this chapter exits the domain of DC networks to focus on the application of the resource disaggregation concept in the fog computing layer of the cloud-of-things architecture. Furthermore, logical disaggregation of TSs is adopted as a representation of the resource disaggregation concept.

6.3 MILP Model for Fog Applications Placement

6.3.1 System Setup

The placement of fog computing applications in traditional and disaggregated fog computing nodes is explored given some constraints. The constraints include resource availability, application performance, energy efficiency and resource utilisation. Although fog applications may require functions across the different tiers of the cloud-of-things architecture, only functions that can be performed in the fog computing tier are considered. The primary aim of fog computing concept is to host application and services on the nearest fog devices. However, this work also assumes that some fog computing sites can be better than others based on energy efficiency. The primary assumption is

that it is better to host fog applications in central locations at the network edge before a local fog node is provisioned for such applications. This is expected to improve overall energy efficiency of the layer.

It is assumed that preferred fog computing nodes in the fog infrastructure are either specialised wired/wireless network equipment which can support generic application as shown in Figure 6.1. Alternatively, existing computing infrastructure owned by an enterprise or a network provider are also adopted as fog computing sites. The preferred fog computing sites may also support mission critical traditional applications i.e., virtual machine (VM) and/or virtual network functions (VNF) required by their owners. Therefore, each traditional app (TA) is associated with specific fog computing sites in the network topology. The spare computing resource capacity in such sites, which is not used to support VM/VNF of TA, is made available to the pool of federated fog computing capacity. The use of both traditional and disaggregated computing infrastructure in such fog sites, which have been integrated into a fog network, are compared.

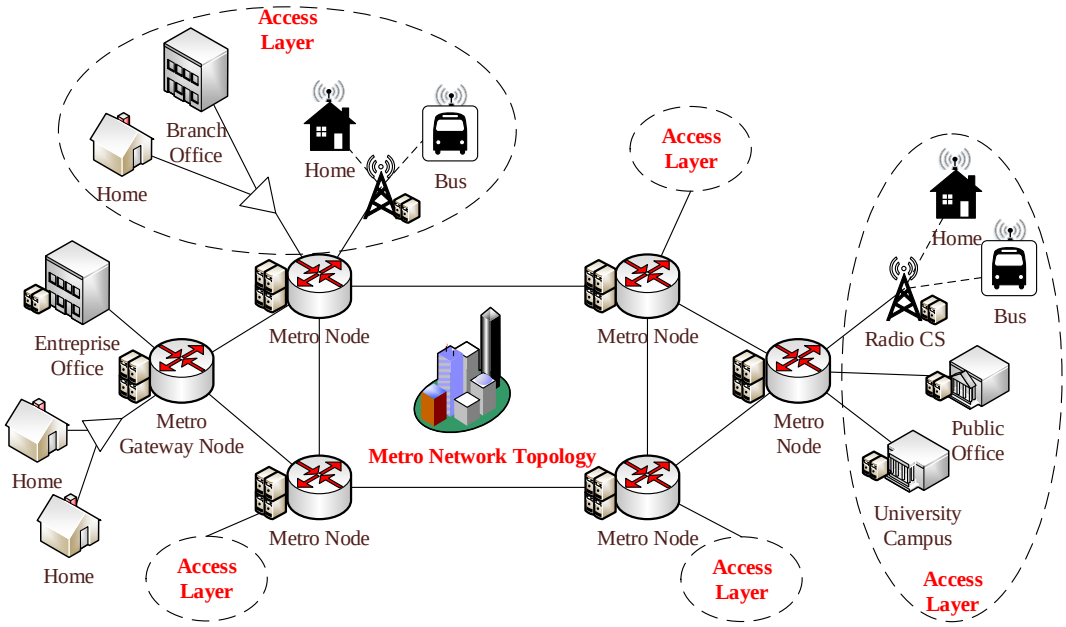


Figure 6.1: Metro fog network

Furthermore, a scenario where the fog computing layer must process all delay sensitive application is considered. This is because such applications cannot be supported by the centralised cloud computing architecture. If a delay sensitive application is not provisioned in the fog network, a local fog node must be provisioned at the source of the request for that application. Every provisioned instance of any fog application leads to corresponding pre-processing and post-processing traffic in the network. Regular network traffic from (to) each access node comprises of traditional network traffic and the

traffic from (to) other applications which are processed in centralised cloud computing node. Only metropolitan area network (MAN) and access network delay is considered in this work. Computation (processing) delay is not considered at each fog node. A scenario where the cumulative computation requirements of all users served by an instance of a given fog application is less than the computing capacity provisioned for that fog application is considered. Hence, minimal computation delay is suffered. Delay estimation considers only link communication delay. This is a sum of propagation and congestion delay experienced on each link in the network topology.

A scenario where fog applications traffic between a fog computing site and end-users follows a single path is considered to simplify delay calculations. Such single low latency path is provisioned by the network service provider to support interactive fog apps created in the fog network. Similar to the work of the authors in [122], a maximum delay threshold is adopted for interactive fog apps during delay-aware placement. The network components traversed by network traffic in the access layer differ according to the different use cases at the access network layer. Figure 6.2 gives illustrations of access layer use cases and their corresponding access network architecture.

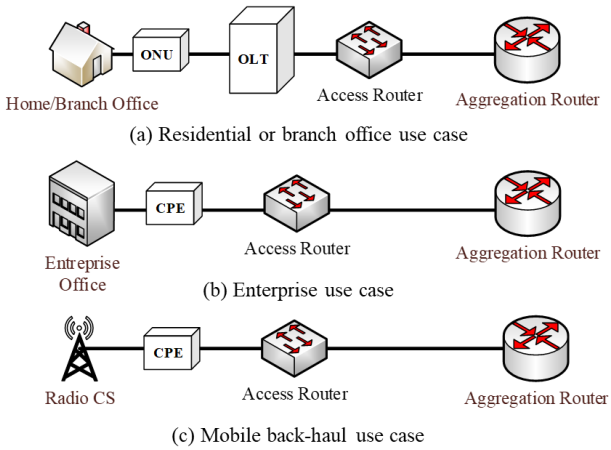


Figure 6.2: Access layer use cases and network architectures in metro fog network

6.3.2 MILP Model Description

In this sub-section, a MILP model that efficiently assigns instances of interactive applications into distributed fog computing nodes within a MAN topology is presented. The model minimises network power consumption, fog computing power consumption and the resulting power consumption of rejected fog applications. The model also minimises the approximated total queuing delay incurred in the network. Given:

- A MAN topology comprising of sets of metro and access network nodes and corresponding inter-connecting physical link capacities as illustrated in Figure 6.1;
- The availability of fog computing capacity in selected fog sites/network nodes in the MAN topology; and
- The locations of clusters of end-users/IoT-devices with explicit demand for an instance of fog applications;

The model determines the number of instances of each fog app that can be provisioned and the optimal location of each provisioned instance while enforcing defined constraints. The data traffic at a given node is proportional to the number of users in that node. Hence, both pre-processing and post processing data traffic of each fog app instance are defined in Gbps per user. Furthermore, only an instance of a given fog app is allocated to all users of that fog app in each access node. The model parameters and variables are given as follows and linear approximations are made as required to ensure linearity.

Network sets and parameters:

N	Set of all network nodes
NB_m	Set of neighbour nodes of network node $m \in N$
U	Set of metro network nodes, $U \subseteq N$
U_m	Set of neighbour metro nodes of metro node $m \in U$
G	Set of gateway nodes in metro network topology, $G \subseteq U$
AN	Set of access network nodes $AN \subseteq N$
AN_a	Set of metro nodes that are neighbours of access network node $a \in AN$
AC_a	$AC_a = 1$ if access network node $a \in AN$ has a consumer premises equipment. Otherwise, $AC_a = 0$.
AP_a	$AP_a = 1$ if access network node $a \in AN$ has a PON ONU. Otherwise, $AP_a = 0$.
PL_{mn}	Bandwidth of physical link (m,n) $m \in N, n \in NB_m$
Δ_{an}	$\Delta_{an} = 1$ if node $n \in N$ is an access network node $a \in AN$. Otherwise, $\Delta_{an} = 0$
RT_{mn}	Regular traffic on physical link (m,n) $m \in N, n \in NB_m$
PD_{mn}	Propagation delay on physical link (m,n) $m \in N, n \in NB_m$

LP_{mn} Set of linear pieces (linear approximations) used to linearise the delay curve of the delay experienced on link (m,n) $m \in N, n \in NB_m$

∇_{mnq} Rate of linear piece $q \in LP_{mn}$ of the linear approximation of the delay experienced on link (m,n) $m \in N, n \in NB_m$

ζ_{mnq} Intercept of linear piece $q \in LP_{mn}$ of the linear approximation of the delay experienced on link (m,n) $m \in N, n \in NB_m$

LU_{mn} Upper bound of queuing delay experienced on link (m,n) $m \in N, n \in NB_m$

CE Consumer premises equipment power consumption

MA Metro Ethernet access switch energy per bit

MG Metro Ethernet aggregation switch energy per bit

NU PON ONU power consumption

OL PON OLT energy per bit

δ Queuing penalty of the network topology in Watt per second.

Fog applications sets and parameters:

F Set of fog apps

TA Set of traditional fog apps, $TA \subseteq F$

E Set of emerging fog apps, $E \subseteq F$

FC_f Compute resource demand of fog app $f \in F$

FM_f Memory resource demand of fog app $f \in F$

FS_f Storage resource demand of fog app $f \in F$

FU_e Uplink data rate per user of emerging fog app $e \in E$

FD_e Downlink data rate per user of emerging fog app $e \in E$

TS_{tn} $TS_{tn} = 1$ if traditional fog app $t \in TA$ is associated with network node $n \in N$. Otherwise, $TS_{tn} = 0$

EG_{en} $EG_{en} = 1$ if node $n \in G$ is the gateway node of emerging fog app $e \in E$. Otherwise, $EG_{en} = 0$

EA_{ea} $EA_{ea} = 1$ if users in node $a \in AN$ make request for an instance of emerging fog app $e \in E$. Otherwise, $EA_{ea} = 0$.

UN_{ea} Number of users in node $a \in AN$ requesting an instance of emerging fog app $e \in E$.

ED_e Emerging fog app $e \in E$ maximum delay threshold.

γ Cost coefficient of power consumed as a result of rejecting traditional fog apps.

$\emptyset$ Cost coefficient of power consumed as a result of rejecting emerging fog apps.

Fog computing nodes sets and parameters:

C Set of CPU resource components.

M Set of memory resource components.

S Set of storage resource components.

$\mathbb{C}_j$ Capacity of CPU component $j \in C$

IC Idle power consumption as a fraction of the maximum CPU power consumption.

CP_j Maximum power consumption of CPU $j \in C$

ΔC_j Power factor of CPU $j \in C$; $\Delta C_j = \frac{CP_j - IC \cdot CP_j}{\mathbb{C}_j}$

$\mathbb{M}_j$ Capacity of memory component $j \in M$

IM Idle power consumption as a fraction of the maximum memory power consumption.

MP_j Maximum power consumption of memory $j \in M$

ΔM_j Power factor of memory $j \in M$; $\Delta M_m = \frac{MP_j - IM \cdot MP_j}{\mathbb{M}_j}$

$\mathbb{S}_j$ Capacity of storage component $j \in S$

IS Idle power consumption as a fraction of the maximum storage power consumption.

SP_j Maximum power consumption of storage $j \in S$

ΔS_j Power factor of storage $j \in S$; $\Delta S_j = \frac{SP_j - IS \cdot SP_j}{\mathbb{S}_j}$

$M\Delta C$ Power factor of CPU component with highest power consumption.

CPM Maximum power consumption of CPU component with highest power consumption.

$M\Delta M$ Power factor of memory component with highest power consumption.

MPM Maximum power consumption of memory component with highest power consumption.

$M\Delta S$ Power factor of storage component with highest power consumption.

SPM Maximum power consumption of storage component with highest power consumption.

A Set of computing nodes

CN_{cx} $CN_{cx} = 1$ if CPU component $c \in C$ is placed in computing node $x \in A$. Otherwise, $CN_{cx} = 0$.

MN_{mx} $MN_{mx} = 1$ if memory component $m \in M$ is placed in computing node $x \in A$. Otherwise, $MN_{mx} = 0$.

SN_{sx} $SN_{sx} = 1$ if storage component $s \in S$ is placed in compute node $x \in A$. Otherwise, $SN_{sx} = 0$.

AM_{xn} $AM_{xn} = 1$ if compute node $x \in A$ is placed in network node $n \in N$. Otherwise, $AM_{xn} = 0$.

Q A large enough number.

Variables:

$\boldsymbol{c_{fc}}$ $\boldsymbol{c_{fc}} = 1$ if an instance of fog app $f \in F$ is in CPU component $c \in C$. Otherwise, $\boldsymbol{c_{fc}} = 0$.

$\boldsymbol{m_{fm}}$ $\boldsymbol{m_{fm}} = 1$ if an instance of fog app $f \in F$ is in memory component $m \in M$. Otherwise, $\boldsymbol{m_{fm}} = 0$.

$\boldsymbol{s_{fs}}$ $\boldsymbol{s_{fs}} = 1$ if an instance of fog app $f \in F$ is in storage component $s \in S$.Otherwise, $\boldsymbol{s_{fs}} = 0$.

$\mathbb{c}_{\boldsymbol{c}}$ $\mathbb{c}_{\boldsymbol{c}} = 1$ if CPU $c \in C$ is active. Otherwise, $\mathbb{c}_{\boldsymbol{c}} = 0$.

$\mathbb{m}_{\boldsymbol{m}}$ $\mathbb{m}_{\boldsymbol{m}} = 1$ if memory $m \in M$ is active. Otherwise,$\mathbb{m}_{\boldsymbol{m}} = 0$.

$\mathbb{s}_{\boldsymbol{s}}$ $\mathbb{s}_{\boldsymbol{s}} = 1$ if storage $s \in S$ is active. Otherwise, $\mathbb{s}_{\boldsymbol{s}} = 0$.

$\mathbb{x}_{\boldsymbol{eca}}$ $\mathbb{x}_{\boldsymbol{eca}} = 1$ if the instance of emerging fog app $e \in E$ in CPU component $c \in C$ is allocated to users in access node $a \in AN$. Otherwise, $\mathbb{x}_{\boldsymbol{eca}} = 0$.

$\mathbb{v}_{\boldsymbol{ea}}$ $\mathbb{v}_{\boldsymbol{ea}} = 1$ if the emerging app $e \in E$ requested by node $a \in AN$ has been provisioned. Otherwise $\mathbb{v}_{\boldsymbol{ea}} = 0$.

$\boldsymbol{\varphi_{esa}}$	$\boldsymbol{\varphi_{esa}} \geq 1$ if an instance of emerging fog app $e \in E$ in node $s \in N$ is allocated to users of that app in access node $a \in AN$. Otherwise, $\boldsymbol{\varphi_{esa}} = 0$.
$\mathbb{k}_{\boldsymbol{sd}}$	Post-processing traffic from instances of all fog apps in network node $s \in N$ to gateway node $d \in G$.
$\mathbb{y}_{\boldsymbol{sdec}}$	Pre-processing traffic from users in node $s \in N$ to node $d \in N$ that hosts an instance of emerging fog app $e \in E$ placed in CPU component $c \in C$ in node $d \in N$.
$\mathbb{z}_{\boldsymbol{sdec}}$	Post-processing traffic from compute node $s \in N$ to users in node $d \in N$. Node $s \in N$ hosts an instance of emerging fog app $e \in E$ placed in CPU component $c \in C$. The instance of emerging fog app $e \in E$ placed in CPU component $c \in C$ was allocated to users in node $d \in N$.
$\boldsymbol{L_{sdec}}$	Traffic from node $s \in N$ to node $d \in N$ due to the presence of emerging fog app $e \in E$ in CPU component $c \in C$.
$\mathbb{h}_{\boldsymbol{mn}}^{\boldsymbol{sd}}$	Volume of $\mathbb{k}_{\boldsymbol{sd}}$ traffic routed on physical link (m, n)
$\boldsymbol{\lambda_{mn}}$	Volume of cloud bound traffic on physical link (m, n).
$\boldsymbol{\mathcal{H}_{mn}^{sdec}}$	Flow of latency sensitive traffic (emerging fog applications traffic) $\boldsymbol{L_{sdec}}$ on physical link (m, n)
$\mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}}$	$\mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} = 1$ if a flow of $\boldsymbol{L_{sdec}}$ is present on physical link (m, n). Otherwise $\mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} = 0$.
$\boldsymbol{\Lambda_{mn}}$	Volume of latency sensitive traffic on physical link (m, n)
$\boldsymbol{\Gamma_{mn}}$	Total traffic on physical link (m, n)
$\mathbb{t}_{\boldsymbol{t}}$	State of traditional fog app $t \in TA$.
$\boldsymbol{TCRTA}$	Total cost of rejected traditional fog apps in Watt.
$\boldsymbol{TCREA}$	Total cost of rejected emerging fog app in Watt.
$\boldsymbol{\alpha_t}$	Power penalty as a result of rejecting traditional fog app $t \in TA$ in Watt.
$\boldsymbol{\beta_e}$	Power penalty as a result of rejecting emerging fog app $e \in E$ in Watt.
$\mathbb{W}_{\boldsymbol{mn}}$	M/M/1 queuing delay experienced on physical link (m, n)

$\boldsymbol{TL_{mn}^{sdec}}$ $\boldsymbol{TL_{mn}^{sdec}}$ is the queuing delay experienced by flow $\boldsymbol{L_{sdec}}$ on physical link (m, n) on the path selected for the flow.

$\boldsymbol{PD_{mn}^{sdec}}$ $\boldsymbol{PD_{mn}^{sdec}}$ is the propagation delay experienced by flow $\boldsymbol{L_{sdec}}$ on physical link (m, n) on the path selected for the flow.

$\boldsymbol{WL_{sdec}}$ Total delay of flow $\boldsymbol{L_{sdec}}$ on all physical links. Sum of congestion in physical links (queuing delay) and propagation delay on physical links on the path.

$\boldsymbol{RD_{sdec}}$ Round trip delay between a node containing users of an emerging fog app and the network node hosting the instance assigned to the users.

$\boldsymbol{TQ}$ Approximated total queuing delay experienced on physical links of the network topology.

Delay related variables in the MILP model are derives as follows.

$$\boldsymbol{PD_{mn}^{sdec}} = PD_{mn}\ \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \tag{6.1}$$

$$\forall\, s, d \in N, \forall\, e \in E, \forall\, c \in C, \forall\, m \in N, \forall\, n \in NB_m$$

Equation (6.1) gives the propagation delay experienced by flow $\boldsymbol{L_{sdec}}$ on physical link (m, n) on the path selected for the flow.

$$\boldsymbol{WL_{sdec}} = \sum_{m \in N} \sum_{n \in NB_m} \left(\boldsymbol{TL_{mn}^{sdec}} + \boldsymbol{PD_{mn}^{sdec}}\right) \tag{6.2}$$

$$\forall\, s, d \in N, \forall\, e \in E, \forall\, c \in C$$

Equation (6.2) gives the total delay experienced by flow $\boldsymbol{L_{sdec}}$ on all physical links. It is a sum of delay experienced due to congestion on physical links and propagation delay on physical links on the path.

$$\boldsymbol{RD_{sdec}} = \boldsymbol{WL_{sdec}} + \boldsymbol{WL_{dsec}} \tag{6.3}$$

$$\forall\, s, d \in N, \forall\, e \in E, \forall\, c \in C$$

Equation (6.3) gives the round-trip delay experienced between a node with users of an emerging fog app and the network node hosting the instance assigned to the users.

$$\boldsymbol{TQ} = \sum_{m \in N} \sum_{n \in NB_m} \mathbb{W}_{\boldsymbol{mn}} \tag{6.4}$$

Equation (6.4) gives the approximated total delay experienced on all physical links of the network topology.

The following equations present the derivation of variables that aid the calculation of network traffic and power consumption.

$$\boldsymbol{\varphi_{esa}} = \sum_{c \in C} \sum_{x \in A} \mathbb{x}_{\boldsymbol{eca}} \, CN_{cx} \, AM_{xs} \tag{6.5}$$

$$\forall\, s \in N, \forall\, a \in AN, \forall\, e \in E$$

Equation (6.5) gives the variable $\boldsymbol{\varphi_{esa}}$ which depends on the placement of emerging fog application. The compute capacity of an instance of emerging fog app $e \in E$ is assumed to be greater than the maximum compute capacity required by the cluster of users of that app in all access node. Hence, $\boldsymbol{\varphi_{esa}} > 1$ is avoided.

$$\mathbb{y}_{\boldsymbol{sdec}} = \sum_{x \in A} \sum_{a \in AN} \mathbb{x}_{\boldsymbol{eca}} \, FU_e \, UA_{ea} \, \Delta_{as} \, CN_{cx} \, AM_{xd} \tag{6.6}$$

$$\forall\, s, d \in N, \forall\, e \in E, \forall\, c \in C$$

$$\mathbb{z}_{\boldsymbol{sdec}} = \sum_{x \in A} \sum_{a \in AN} \mathbb{x}_{\boldsymbol{eca}} \, FD_e \, UA_{ea} \, \Delta_{ad} \, CN_{cx} \, AM_{xs} \tag{6.7}$$

$$\forall\, s, d \in N, \forall\, e \in E, \forall\, c \in C$$

$$\mathbb{k}_{\boldsymbol{sd}} = \sum_{a \in AN} \sum_{e \in E} \boldsymbol{\varphi_{esa}} \, FD_e \, UA_{ea} \, EG_{ed} \tag{6.8}$$

$$\forall\, s \in N, \forall\, d \in G$$

Equations (6.6) and (6.7) give the pre-processing and post-processing traffic respectively between users in access nodes and instances of emerging fog applications placed in access and metro nodes. Equation (6.8) gives the post-processing traffic between instances of emerging fog applications placed in access or metro nodes and gateway metro nodes.

$$\boldsymbol{L_{sdec}} = \mathbb{y}_{\boldsymbol{sdec}} + \mathbb{z}_{\boldsymbol{sdec}} \tag{6.9}$$

$$\forall\, s, d \in N, \forall\, e \in E, \forall\, c \in C$$

Equation (6.9) gives the total traffic between a pair of nodes due to the presence of an instance of an emerging fog app. It is a sum of the pre-processing and post-processing traffic.

$$\boldsymbol{\Lambda_{mn}} = \sum_{s \in N} \sum_{d \in N} \sum_{e \in E} \sum_{c \in C} \boldsymbol{\mathcal{H}_{mn}^{sdec}} \tag{6.10}$$

$$\forall\, m \in N, \forall\, n \in NB_m$$

Equation (6.10) gives the latency sensitive traffic routed over a physical link by summing all the latency sensitive flows over the link.

$$\boldsymbol{\lambda_{mn}} = \sum_{s \in N} \sum_{d \in N} \mathbb{h}_{\boldsymbol{mn}}^{\boldsymbol{sd}} + RT_{mn} \tag{6.11}$$

$$\forall\, m \in N, \forall\, n \in NB_m$$

Equation (6.11) gives the latency tolerant traffic routed over a physical link by summing all the latency tolerant flows over the link and the given regular traffic on that physical link.

$$\boldsymbol{\Gamma_{mn}} = \boldsymbol{\Lambda_{mn}} + \boldsymbol{\lambda_{mn}} \tag{6.12}$$

$$\forall\, m \in N, \forall\, n \in NB_m$$

Equation (6.12) gives the total traffic over a link by summing latency sensitive and latency tolerant traffic routed over the link.

Total network power consumption $\boldsymbol{TNPC}$ is given as

$$\boldsymbol{TNPC} = \boldsymbol{TANP} + \boldsymbol{TMNP} \tag{6.13}$$

where $\boldsymbol{TANP}$ is the total access network power consumption and is given by

$$\boldsymbol{TANP} = \sum_{a \in AN} (CE\ AC_a + NU\ AP_a) + \sum_{a \in AN} \sum_{m \in AN_a} (\boldsymbol{\Gamma_{am}} + \boldsymbol{\Gamma_{ma}})(MA + AP_a\ OL) \tag{6.14}$$

and $\boldsymbol{TMNP}$ is the total metro network power consumption and is given as

$$\boldsymbol{TMNP} = \sum_{m \in U} (\mathbb{u}_{\boldsymbol{m}} + \mathbb{g}_{\boldsymbol{m}} + \mathbb{q}_{\boldsymbol{m}})\, MG \tag{6.15}$$

where $\mathbb{u}_m$ is the traffic relayed by a metro node m and is given as,

$$\mathbb{u}_{\boldsymbol{m}} = \sum_{s \in N} \sum_{d \in N} \sum_{n \in NB_m} \mathbb{h}_{\boldsymbol{mn}}^{\boldsymbol{sd}} + \sum_{n \in NB_m} \sum_{s \in N} \sum_{d \in N} \sum_{e \in E} \sum_{c \in C} \boldsymbol{\mathcal{H}}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \tag{6.16}$$

$$\forall\, m \in U, s, d \in N, s \neq m, d \neq m$$

where $\mathbb{g}_{\boldsymbol{m}}$ is the traffic received by a metro node m and is given as,

$$\mathbb{g}_{\boldsymbol{m}} = \sum_{s \in N} \sum_{n \in NB_m} \mathbb{h}_{\boldsymbol{nm}}^{\boldsymbol{sm}} + \sum_{s \in N} \sum_{n \in NB_m} \sum_{e \in E} \sum_{c \in C} \boldsymbol{\mathcal{H}}_{\boldsymbol{nm}}^{\boldsymbol{smec}} + \sum_{n \in NB_m} RT_{nm} \tag{6.17}$$

$$\forall\, m \in U$$

and finally, $\mathbb{q}_{\boldsymbol{m}}$ is the traffic transmitted by a metro node m and is given as,

$$\mathbb{q}_{\boldsymbol{m}} = \sum_{d \in G} \sum_{n \in NB_m} \mathbb{h}_{\boldsymbol{mn}}^{\boldsymbol{md}} + \sum_{d \in N} \sum_{n \in NB_m} \sum_{e \in E} \sum_{c \in C} \boldsymbol{\mathcal{H}}_{\boldsymbol{mn}}^{\boldsymbol{mdec}} + \sum_{n \in NB_m} RT_{mn} \tag{6.18}$$

$$\forall\, m \in U$$

Total fog computing power consumption ($\boldsymbol{TFPC}$) is given by:

$$\boldsymbol{TFPC} = \boldsymbol{TCPC} + \boldsymbol{TMPC} + \boldsymbol{TSPC} \tag{6.19}$$

where $\boldsymbol{TCPC}$ is the total power consumption of CPU resources in fog network and is given as:

$$\boldsymbol{TCPC} = \sum_{c \in C} \left((IC\ CP_c\ \mathbb{c}_{\boldsymbol{c}}) + \sum_{f \in F} \Delta C_c\ \boldsymbol{c_{fc}}\ FC_f \right) \tag{6.20}$$

$\boldsymbol{TMPC}$ is the total power consumption of memory resources in fog network and is given as:

$$\boldsymbol{TMPC} = \sum_{m \in M} \left((\mathrm{IM}\ MP_m\ \mathbb{m}_{\boldsymbol{m}}) + \sum_{f \in F} \Delta M_m\ \boldsymbol{m_{fm}}\ FM_f \right) \tag{6.21}$$

and $\boldsymbol{TSPC}$ is the total power consumption of storage resources in fog network and is given as:

$$\boldsymbol{TSPC} = \sum_{s \in S} \left((\mathrm{IS}\ SP_s\ \mathbb{s}_{\boldsymbol{s}}) + \sum_{f \in F} \Delta S_s\ \boldsymbol{s_{fs}}\ FS_f \right) \tag{6.22}$$

The total cost of rejected traditional fog apps in the distributed fog network is given as

$$\boldsymbol{TCRTA} = \sum_{t \in TA} \boldsymbol{\alpha_t} \tag{6.23}$$

where,

$$\boldsymbol{\alpha_t} = \big((IC\ CPM + M\Delta C\ FC_t) + (IM\ MPM + M\Delta M\ FM_t) + (IS\ SPM + M\Delta S\ FD_t)\big)\mathbb{t}_{\boldsymbol{t}} \tag{6.24}$$

$$\forall\ t\ \in TA$$

where the state of traditional application t is given by:

$$\mathbb{t}_{\boldsymbol{t}} = \sum_{c \in C} (1 - \boldsymbol{c_{tc}}) \tag{6.25}$$

$$\forall\ t\ \in TA$$

The total cost of rejected emerging fog apps in the distributed fog network is given as

$$\boldsymbol{TCREA} = \sum_{e \in E} \boldsymbol{\beta_e} \tag{6.26}$$

where

$$\boldsymbol{\beta_e} = \big((IC\ CPM + M\Delta C\ FC_e) + (IM\ MPM + M\Delta M\ FM_e) + (IS\ SPM + M\Delta S\ FS_e)\big) \sum_{a \in AN} (1 - \mathbb{v}_{\boldsymbol{ea}}) \quad (6.27)$$

$$\forall\ e\ \in E$$

where $\mathbb{v}_{\boldsymbol{ea}}$ indicates if the emerging application e requested by node a has been provisioned or not, and is given by

$$\mathbb{v}_{\boldsymbol{ea}} = \sum_{c \in C} \mathbb{x}_{\boldsymbol{eca}} \quad (6.28)$$

$$\forall\ e\ \in E, \forall\ a \in\ AN$$

Note that the cost of rejecting an app is defined as the maximum power consumed if it is accepted, i.e., the power consumption in the case where inactive components must be turned-on to support the app.

The model is defined as follows:

Objective: Minimise

$$\boldsymbol{TNPC + TFPC} + \gamma\, \boldsymbol{TCRTA} + \emptyset\, \boldsymbol{TCREA} + \delta\, \boldsymbol{TQ} \quad (6.29)$$

The objective of the model is to minimise a weighted sum of the total network power consumption, total fog power consumption and the total cost of rejected traditional, emerging fog applications and the cost of approximated total delay in the network as given by the expression in (6.29). Setting γ to a high value ensures that the total cost of rejected traditional fog apps is significantly higher than the cost of rejected emerging fog apps. Hence, higher priority is given to provisioning traditional fog apps in the objective function. $\emptyset$ can also be varied to increase or decrease the cost of rejected emerging fog apps. The value of δ dictates the weight of approximated total queuing delay in the objective function. $\delta \ll 1$ represents a network with trivial queuing delay penalty while $\delta \gg 1$ represents a network with significant queuing delay penalty.

Subject to:

Fog DC related constraints

$$\sum_{c \in C} \boldsymbol{c_{fc}} = \sum_{m \in M} \boldsymbol{m_{fm}} \quad (6.30)$$

$$\forall\ f\ \in F$$

$$\sum_{c \in C} \boldsymbol{c_{fc}} = \sum_{s \in S} \boldsymbol{s_{fs}} \quad (6.31)$$

$$\forall\ f\ \in F$$

Constraints (6.30) and (6.31) ensure that the number of instances of CPU resources provisioned for any (traditional or emerging) fog app is equal to the number of instances of memory and storage resources provisioned for that app across the distributed fog network.

$$\sum_{c \in C} \boldsymbol{c_{fc}}\, CN_{cx} \;=\; \sum_{m \in M} \boldsymbol{m_{fm}}\, MN_{mx} \tag{6.32}$$

$$\forall\, f \;\in F, \forall\, x \;\in A$$

$$\sum_{c \;\in C} \boldsymbol{c_{fc}}\, CN_{cx} \;=\; \sum_{s \in S} \boldsymbol{s_{fs}}\, SN_{sx} \tag{6.33}$$

$$\forall\, f \;\in F, \forall\, x \;\in A$$

Constraints (6.32) and (6.33) are the locality constraints when the traditional server architecture is adopted in compute nodes. They ensure that the CPU, memory, and storage components used to provision a given instance of a fog app are in the same compute nodes.

$$\sum_{x \in A} \sum_{c \in C} \boldsymbol{c_{fc}}\, CN_{cx}\, AM_{xn} \;=\; \sum_{x \in A} \sum_{m \in M} \boldsymbol{m_{fm}}\; MN_{mx}\, AM_{xn} \tag{6.34}$$

$$\forall\, f \;\in F, \forall\, n \in N$$

$$\sum_{x \in A} \sum_{c \in C} \boldsymbol{c_{fc}}\, CN_{cx}\, AM_{xn} \;=\; \sum_{x \in A} \sum_{s \in S} \boldsymbol{s_{fs}}\, SN_{sx}\, AM_{xn} \tag{6.35}$$

$$\forall\, f \;\in F, \forall\, n \in N$$

Constraints (6.34) and (6.35) are the locality constraints when the disaggregated server architecture is adopted in compute nodes. They ensure that the CPU, memory, and storage components used to provision a given instance of a fog app are in the same network node but not necessarily in the same compute node.

$$\sum_{x \in A} \sum_{c \in C} \boldsymbol{c_{tc}}\, CN_{cx}\, AM_{xn} \;= TS_{tn} \tag{6.36}$$

$$\forall\, t \;\in TA, \forall\, n \in N$$

Constraint (6.36) is the workload locality constraint for a traditional fog app that is associated with a given network node.

$$\sum_{c \in C} \boldsymbol{c_{fc}}\, CN_{cx} \;\leq 1 \tag{6.37}$$

$$\forall\, f \;\in F, \forall\, x \in A$$

$$\sum_{m \in M} \boldsymbol{m_{fm}}\, MN_{mx} \;\leq 1 \tag{6.38}$$

$$\forall f \in F, \forall x \in A$$

$$\sum_{s \in S} \boldsymbol{s_{fs}} SN_{sx} \leq 1 \tag{6.39}$$

$$\forall f \in F, \forall x \in A$$

Constraints (6.37) - (6.39) are SLA constraints which ensure robustness of the fog network. They ensure that only an instance of fog app f is provisioned within a given compute node. Hence, the impact of a compute node failure is minimised for a fog app with multiple instances.

$$\sum_{f \in F} FC_f \, \boldsymbol{c_{fc}} \leq \mathbb{C}_c \tag{6.40}$$

$$\forall c \in C$$

$$\sum_{f \in F} FM_f \, \boldsymbol{m_{fm}} \leq \mathbb{M}_m \tag{6.41}$$

$$\forall m \in M$$

$$\sum_{f \in F} FS_f \, \boldsymbol{s_{fs}} \leq \mathbb{S}_s \tag{6.42}$$

$$\forall s \in S$$

Constraints (6.40) - (6.42) denotes resource capacity constraints for each CPU, memory, and storage component in the fog network. They ensure that each resource component capacity is not exceeded.

$$\sum_{f \in F} \boldsymbol{c_{fc}} \geq \mathbb{c}_{\boldsymbol{c}} \tag{6.43}$$

$$\forall c \in C$$

$$\sum_{f \in F} \boldsymbol{c_{fc}} \leq Q \, \mathbb{c}_{\boldsymbol{c}} \tag{6.44}$$

$$\forall c \in C$$

$$\sum_{f \in F} \boldsymbol{m_{fm}} \geq \mathbb{m}_{\boldsymbol{m}} \tag{6.45}$$

$$\forall m \in M$$

$$\sum_{f \in F} \boldsymbol{m_{fm}} \leq Q \, \mathbb{m}_{\boldsymbol{m}} \tag{6.46}$$

$$\forall m \in M$$

$$\sum_{f \in F} \boldsymbol{s_{fs}} \geq \mathbb{s}_{\boldsymbol{s}} \tag{6.47}$$

$$\forall\, s \in S$$

$$\sum_{f \in F} \boldsymbol{\mathcal{s}_{fs}} \leq Q\, \mathbb{s}_{\boldsymbol{s}} \tag{6.48}$$

$$\forall\, s \in S$$

Constraints (6.43) - (6.48) derive the state of CPU, memory, and storage components.

Fog app instance related constraints

$$\sum_{c \in C} \mathbb{x}_{\boldsymbol{eca}} \leq 1 \tag{6.49}$$

$$\forall\, e \in E, \forall\, a \in AN$$

Constraint (6.49) ensures that the cluster of users requesting for an emerging fog app is assigned to at most one instance of that app when feasible.

$$\sum_{a \in AN} \mathbb{x}_{\boldsymbol{eca}} \geq \boldsymbol{c_{ec}} \tag{6.50}$$

$$\forall\, c \in C, e \in E$$

$$\sum_{a \in AN} \mathbb{x}_{\boldsymbol{eca}} \leq Q\, \boldsymbol{c_{ec}} \tag{6.51}$$

$$\forall\, c \in C, e \in E$$

Constraints (6.50) and (6.51) ensure that each instance of an emerging fog app in a CPU component is allocated to one or more user clusters in access nodes. Otherwise, the instance should not be created.

$$\mathbb{x}_{\boldsymbol{eca}} = EA_{ea}\, \mathbb{v}_{\boldsymbol{ea}}\, \boldsymbol{c_{ec}} \tag{6.52}$$

$$\forall\, c \in C, e \in E, a \in AN$$

$$\mathbb{x}_{\boldsymbol{eca}} \leq EA_{ea}\, \boldsymbol{c_{ec}} \tag{6.53}$$

$$\forall\, c \in C, e \in E, a \in AN$$

$$\mathbb{x}_{\boldsymbol{eca}} \leq EA_{ea}\, \mathbb{v}_{\boldsymbol{ea}} \tag{6.54}$$

$$\forall\, c \in C, e \in E, a \in AN$$

$$\mathbb{x}_{\boldsymbol{eca}} \geq EA_{ea}(\mathbb{v}_{\boldsymbol{ea}} + \boldsymbol{c_{ec}}) - 1 \tag{6.55}$$

$$\forall\, c \in C, e \in E, a \in AN$$

Constraint (6.52) derives $\mathbb{x}_{\boldsymbol{eca}}$ which gives the instance of an emerging fog app in a CPU that is assigned to users of that fog app in an access node. $\mathbb{x}_{\boldsymbol{eca}} = 1$ if and only if users of an emerging fop app are present in an access node, an instance of that fog app is in a CPU component and that instance

has been assigned to users of that fog app in access node. Constraints (6.53) - (6.55) implement the product of parameter and variables as illustrated in Constraint (6.52).

Network related constraints

$$\sum_{n \in NB_m} \mathbb{h}_{\boldsymbol{mn}}^{\boldsymbol{sd}} - \sum_{n \in NB_m} \mathbb{h}_{\boldsymbol{nm}}^{\boldsymbol{sd}} = \begin{cases} \mathbb{k}_{\boldsymbol{sd}} & m = s \\ -\mathbb{k}_{\boldsymbol{sd}} & m = d \\ 0 & otherwise \end{cases} \tag{6.56}$$

$$\forall\, s, m \in N, \forall\, d \in G : s \neq d$$

Constraint (6.56) enforces flow conservation for post-processing traffic to the cloud in the physical layer of the network.

$$\sum_{n \in NB_m} \boldsymbol{\mathcal{H}}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} - \sum_{n \in NB_m} \boldsymbol{\mathcal{H}}_{\boldsymbol{nm}}^{\boldsymbol{sdec}} = \begin{cases} \boldsymbol{L}_{\boldsymbol{sdec}} & m = s \\ -\boldsymbol{L}_{\boldsymbol{sdec}} & m = d \\ 0 & otherwise \end{cases} \tag{6.57}$$

$$\forall\, s, d, m \in N, e \in E, c \in C : s \neq d$$

Constraint (6.57) enforces flow conservation for latency sensitive flows in the physical layer of the network.

$$\boldsymbol{\mathcal{H}}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \geq \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \tag{6.58}$$

$$\forall\, s, d, m, n \in N, e \in E, c \in C : s \neq d$$

$$\boldsymbol{\mathcal{H}}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \leq \boldsymbol{Q}\, \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \tag{6.59}$$

$$\forall\, s, d, m, n \in N, e \in E, c \in C : s \neq d$$

Constraints (6.58) and (6.59) give the binary equivalent of $\boldsymbol{\mathcal{H}}_{\boldsymbol{mn}}^{\boldsymbol{sdec}}$.

$$\sum_{n \in NB_m} \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \leq 1 \tag{6.60}$$

$$\forall\, s, d, m \in N, e \in E, c \in C : s \neq d$$

Constraint (6.60) ensures that the flow $\boldsymbol{L}_{\boldsymbol{sdec}}$ is not bifurcated over multiple paths.

$$\boldsymbol{\Gamma}_{\boldsymbol{mn}} \leq PL_{mn} \tag{6.61}$$

$$\forall\, m \in N, n \in NB_m$$

Constraint (6.61) enforces capacity constraint on each physical link (m, n).

Network delay related constraints

$$\boldsymbol{TL}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} = \mathbb{W}_{\boldsymbol{mn}}\, \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \tag{6.62}$$

$$\forall\, s, d, m \in N, n \in NB_m, e \in E, c \in C : s \neq d$$

$$\boldsymbol{TL}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \leq LU_{mn} \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}} \tag{6.63}$$

$$\forall\, s, d, m \in N, n \in NB_m, e \in E, c \in C: s \neq d$$

$$\boldsymbol{TL_{mn}^{sdec}} \leq \mathbb{W}_{\boldsymbol{mn}} \tag{6.64}$$

$$\forall\, s, d, m \in N, n \in NB_m, e \in E, c \in C: s \neq d$$

$$\boldsymbol{TL_{mn}^{sdec}} \geq \mathbb{W}_{\boldsymbol{mn}} - LU_{mn}\,(1 - \mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}}) \tag{6.65}$$

$$\forall\, s, d, m \in N, n \in NB_m, e \in E, c \in C: s \neq d$$

Constraint (6.62) estimates the queuing delay experienced by flow $\boldsymbol{L_{sdec}}$ on physical link(m, n). It is given by the product of $\mathbb{W}_{\boldsymbol{mn}}$ and $\mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}}$. Constraints (6.63) - (6.65) linearise Constraint (6.62). LU_{mn} is the upper bound of the queuing delay experienced on each physical link, it is required to ensure that the delay experienced on a physical link does not exceed a predefined threshold. Given $\mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}}$, Table 6.1 gives the evaluation of constraints (6.63) - (6.65) and the values of $\boldsymbol{TL_{mn}^{sdec}}$ that satisfies the defined constraints.

Table 6.1: Evaluation of constraint (6.63) - (6.65)

$\mathbb{H}_{\boldsymbol{mn}}^{\boldsymbol{sdec}}$	**(6.63)**	**(6.64)**	**(6.65)**	$\boldsymbol{TL_{mn}^{sdec}}$
0	$\boldsymbol{TL_{mn}^{sdec}} \leq 0$	$\boldsymbol{TL_{mn}^{sdec}} \leq \mathbb{W}_{\boldsymbol{mn}}$	$\boldsymbol{TL_{mn}^{sdec}} \geq \mathbb{W}_{\boldsymbol{mn}} - LU_{mn}$	$\boldsymbol{TL_{mn}^{sdec}} = 0$
1	$\boldsymbol{TL_{mn}^{sdec}} \leq LU_{mn}$	$\boldsymbol{TL_{mn}^{sdec}} \leq \mathbb{W}_{\boldsymbol{mn}}$	$\boldsymbol{TL_{mn}^{sdec}} \geq \mathbb{W}_{\boldsymbol{mn}}$	$\boldsymbol{TL_{mn}^{sdec}} = \mathbb{W}_{\boldsymbol{mn}}$

$$\mathbb{W}_{\boldsymbol{mn}} \geq \nabla_{mnq}\, \boldsymbol{\Gamma_{mn}} + \zeta_{mnq} \tag{6.66}$$

$$\forall\, m \in N, n \in NB_m, q \in LP_{mn}$$

Constraint (6.66) represent piecewise linear approximation of queuing delay experienced on physical link (m, n). This is because M/M/1 delay is a non-linear function.

$$\boldsymbol{RD_{sdec}} \leq ED_e \tag{6.67}$$

$$\forall\, s, d \in N, e \in E, c \in C: s \neq d$$

Constraint (6.67) represents the round-trip delay constraint for an emerging fog app e. The round-trip delay experienced by an emerging fog app e must not exceed the predefined threshold.

6.4 Evaluation and Results

6.4.1 Evaluation Scenarios and Input Parameters

The MILP model described in the previous section is used to study the impact of adopting DS architecture in the fog computing layer of the cloud-of-things continuum relative to the use of TS architecture. To minimise execution time

of the MILP model which grows as the size and complexity of the problem increases, a small network topology is considered. The network topology comprises of 4 (metro) central offices (COs) and 16 access nodes. The access nodes include 4 radio cell sites (CSs), 4 enterprise offices (EOs) and 8 homes. Connected to each metro CO are an EO, a radio CS and two residential houses as illustrated in Figure 6.3. The COs and EOs are connected to the metro ring via 40 Gbps links. The homes are connected to the metro ring via 40 Gbps Next-Generation Passive Optical Network 2 (NG-PON2) links. Using 40 Gbps last mile links between metro and access network nodes ensures that network bottlenecks are avoided. Network bottlenecks can lead to the rejection of emerging fog apps.

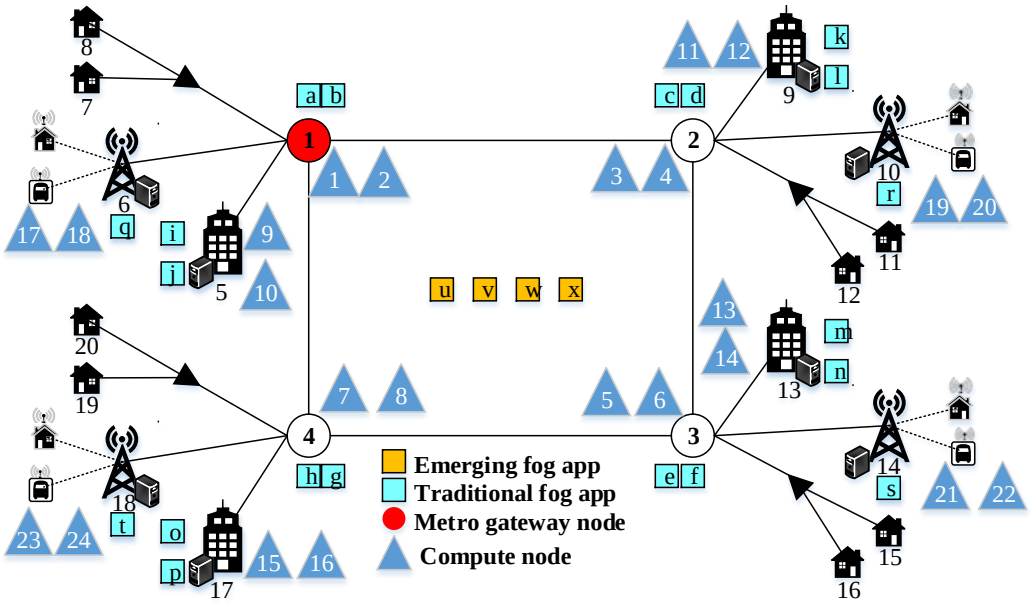


Figure 6.3: Fog Network System Setup

The figure shows the evaluated network topology, the preferred fog computing sites, and the allocation of compute nodes to the fog sites. The figure also shows the traditional fog apps associated with each fog computing site and un-provisioned emerging fog apps.

To further maintain simplicity, the evaluation scenario allocates two servers to each fog computing site. The fog computing sites comprise of metro COs, EOs, and radio CSs in the network topology as illustrated in Figure 6.3. When the TS architecture is adopted, the utilisation scope of each server's intrinsic resource components is limited to that server. On the other hand, when the DS architecture is adopted, servers are logically disaggregated. This expand the utilisation scope of the intrinsic resources of each server at fog computing sites. However, access to disaggregated resource components is limited to the corresponding fog computing site hosting each component. A common configuration is adopted for all servers distributed across the metro

network topology. Each server comprises of one CPU, one memory and one storage resource components. The characteristics of each compute component used to evaluate the MILP model are given in Table 6.2. The power consumption profile of each compute component comprises of an idle portion and another portion that is linearly dependent on the component's utilisation. Servers are not allocated to residential houses because reduction of the number of computing resources deployed at the extreme edge of the network is desired. Moreover, it is unusual to have large computing capacity in residential houses.

Table 6.2: Component capacity and power features

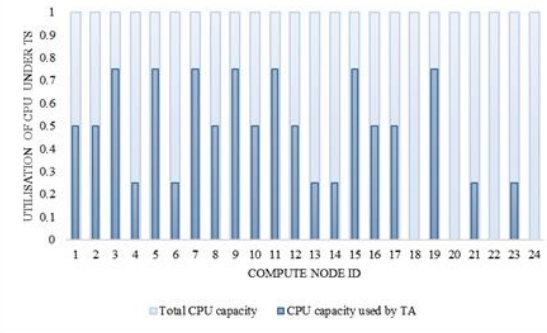
Parameter	**Value**
CPU capacity, CPU peak power, and idle fraction of peak CPU power	3.6 GHz, 130 W and 0.7
Memory capacity, memory peak power, and idle fraction of peak memory power	32 GB, 11.85 W and 0.7
Storage capacity, storage peak power, and idle fraction of peak storage power	320 GB, 6.19 W and 0.7
Metro Ethernet CPE (on-off)	75 W [123]
Metro Ethernet aggregation router (load proportional)	0.9 W/Gb [124]
Metro Ethernet access router (load proportional)	0.243 W/Gb
PON optical line terminal (load proportional)	1.75 W/Gb
PON optical network unit (on-off)	15 W [125]

Each designated fog computing site within the federation of fog computing nodes has one or two in-situ VM/VNFs for mission critical traditional applications (TAs). As illustrated in Figure 6.3, the TAs must be provisioned at the corresponding node. Each CO node has 2 VNFs; each EO has 2 VMs; and each radio CS has 1 VNF. The resource demand of each mission critical TA is illustrated in Table 6.3. Figure 6.4 and Figure 6.5 show the corresponding utilisation of each compute components across distributed fog nodes after optimal placement of mission critical TAs. The results obtained under TS and DS architectures are a solution to the MILP model. The MILP model optimally places TAs without considering emerging fog apps. Figure

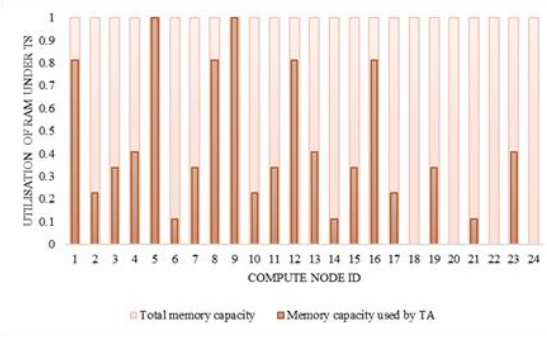
6.4 and Figure 6.5 show the presence of unused computing capacity after TAs have been provisioned under both TS and DS architectures. This spare compute capacity can be used to support emerging fog applications in preferred fog computing sites. Relative to the TS architecture, the DS architecture has greater average active resource utilisation and reduced number of active resource components across fog computing sites. It is expected that these advantages of DS architecture over TS architecture will be maintained when placement of emerging fog apps is considered in parallel with mission critical TAs. However, to achieve optimal efficiency and minimal rejection of emerging fog apps, the placement of each TA within each network node may be revised. Furthermore, the analysis of application placement is focused on emerging fog applications alone. This is because the placement of mission critical TAs is fixed to specific fog computing sites.

Four types of emerging fog applications, which have distributed users in the access layer, are considered. It is assumed that all applications required by each enterprise are either hosted locally as VMs or hosted remotely in cloud DCs. Hence, distributed users of emerging fog applications are not associated with EOs. All user demands for a fog app in each access node are grouped together to form a cluster of user demand in that node. Each radio CS has a group of 5 end-users which collectively form a single clustered demand for each emerging fog app. The users are attached to the radio CS via wireless media. A single end-user located in each residential house, this user forms a single clustered demand for each emerging fog app at the corresponding network node.

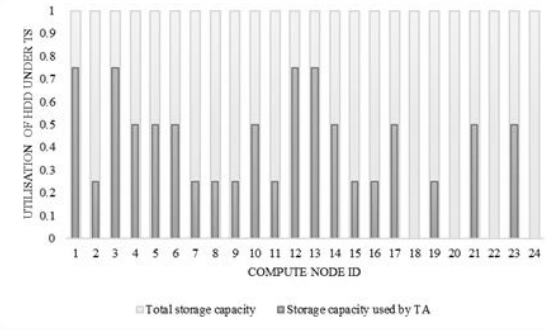
Both VM/VNF of traditional fog apps and emerging fog apps have a mix of resource intensity as illustrated in Table 6.3. Relative to the maximum compute resource capacity some apps are CPU intensive while others are memory intensive. Relative to the capacity of computes resource adopted, fog app “U” has medium CPU demand, high memory demand and low storage demand. Fog app “V” has high CPU demand, high memory demand and medium storage demand. Fog app “W” has medium CPU demand, low memory demand, and low storage demand. Fog app “X” has low CPU, memory, and storage demands. Relative to other emerging fog apps, fog apps U and X have the highest pre and post processing traffic per user as illustrated in Table 6.3.



(a) CPU component utilisation

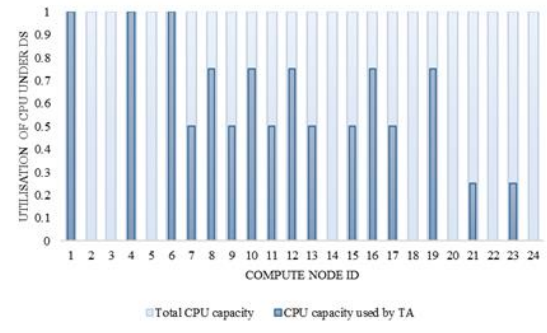


(b) RAM component utilisation

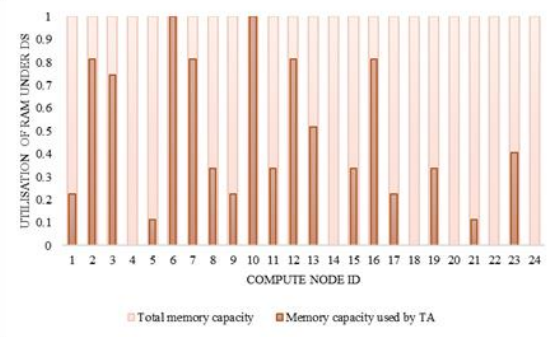


(c) HDD component utilisation

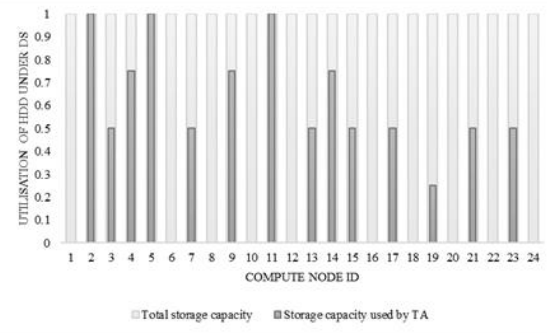
Figure 6.4: Resource component utilisation under TS architecture



(a) CPU component utilisation



(b) RAM component utilisation



(c) HDD component utilisation

Figure 6.5: Resource component utilisation under DS architecture

The placement of apps in centralised cloud DCs is not explicitly considered. However, the impact of cloud destined traffic on the overall performance of metro and access network tiers is considered. It is assumed that traffic to and from the centralise cloud DC is part of the regular traffic traversing the network topology. Priority is given to traffic of emerging fog apps in the fog network but the traffic of other applications and services contribute to the regular traffic traversing metro and access networks. The range of regular traffic on the metro ring and access links are 114 – 120 Gbps and 4 – 5 Gbps respectively. Therefore, regular traffic utilises about 60% and 12.5% of the capacity of a metro and access link respectively. The pre-processing and post-processing data rates per user for each emerging fog app considered are given in Table 6.3.

Table 6.3: Resource demand of fog apps

App Symbol	CPU demand (GHz)	Memory demand (GB)	Storage demand (GB)	Pre-processing data rate per user (Gbps)	Post-processing data rate per user (Gbps)
A	1.8	7.2	80	-	-
B	1.8	26	240	-	-
C	2.7	10.8	240	-	-
D	0.9	13	160	-	-
E	0.9	3.6	160	-	-
F	2.7	32	160	-	-
G	1.8	26	80	-	-
H	2.7	10.8	80	-	-
I	2.7	32	80	-	-
J	1.8	7.2	160	-	-
K	1.8	26	240	-	-
L	2.7	10.8	80	-	-
M	0.9	3.6	160	-	-
N	0.9	13	240	-	-
O	2.7	10.8	80	-	-
P	1.8	26	80	-	-
Q	1.8	7.2	160	-	-
R	2.7	10.8	80	-	-
S	0.9	3.6	160	-	-
T	0.9	13	160	-	-
U	1.8	26	120	0.9	0.45
V	2.7	32	160	0.11	0.05
W	1.8	7.2	80	0.83	0.41
X	0.9	3.6	40	0.43	0.22

After processing at the optimal location, processed data is sent to the requesting users at the edge of the network and to the central cloud for further/historical analysis and persistent storage. It is assumed that the ratio of post-processing data rate to pre-processing data rate is about 50% as illustrated in Table 6.3. In the worst-case scenario, if the request made by a user in node 7 for all emerging fog apps is satisfied about 6% of the access link capacity will be utilised. Recall that Node 7 is a residential house with a single user for each emerging fog app. On the other hand, if the requests made by all users in node 6 are satisfied, about 30% of the access link capacity will be utilised. Node 6 is a radio CS with multiple users for each emerging fog app. Generally, compared to the fog related traffic, regular traffic is dominant in the network topology.

Shared network elements such as Ethernet access and aggregation routers and optical line terminals (OLTs) are assumed to have a load proportional power profile. On the other hand, dedicated network components such as consumer premises equipment (CPE) and optical network units (ONUs) have an on-off power profile. Table 6.2 shows the power profile of each network element and their corresponding values. The exponential M/M/1 delay graph of each network link is divided into 6 linear pieces to implement piecewise linearisation of the non-linear delay function. Figure 6.6 and Figure 6.7 give the piecewise linear approximation of both 200 Gbps and 40 Gbps network links using the 6 linear pieces. The predefined upper bound for link load on both 200 Gbps and 40 Gbps network links are 195 Gbps and 39 Gbps respectively. Both values enforce a corresponding upper bound for queuing delay on each link. These values maybe varied based on desired network performance on the corresponding link.

This study evaluates the energy efficient placement of delay sensitive emerging fog applications in the presence of mission critical traditional fog applications in a shared distributed fog network. Analysis of results from the model focuses on metrics such as TFPC, TNPC, number of fog app instances created, roundtrip delay experienced by users of emerging fog applications. Other evaluation metrics include the number of active resource components across all fog computing sites in fog network and the corresponding average utilisation of each active component type across the fog network. To obtain optimal results, the results show that the MILP model effectively bin-packs workloads demands onto fog computing resources. Bin-packing attempts to achieve optimal resource power and utilisation efficiencies within capacity constraints and limited resource utilisation scope.

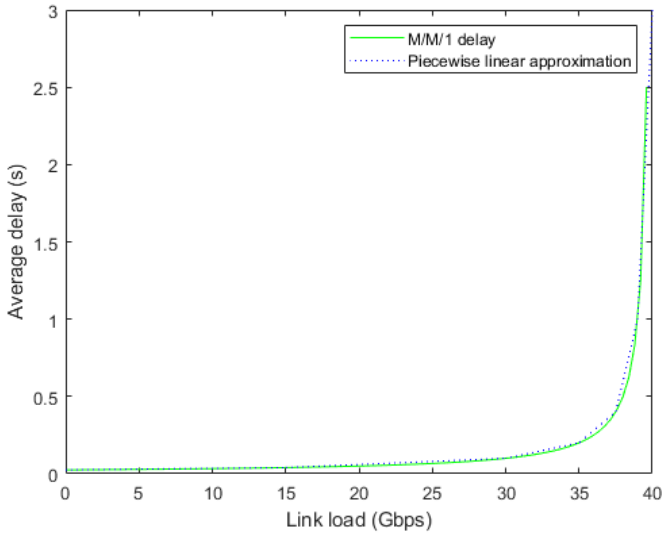


Figure 6.6: M/M/1 average delay experienced on 40 Gbps access link

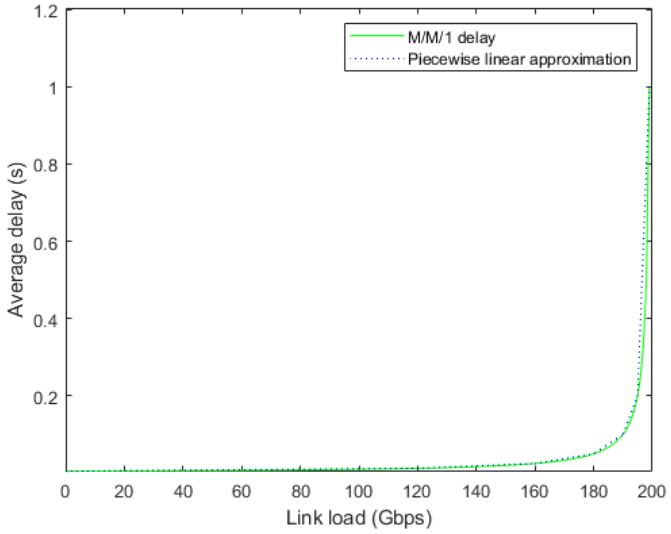


Figure 6.7: M/M/1 average delay experienced on 200 Gbps metro ring link

6.4.2 Energy Efficient Placement under Low Delay Penalty

Under this scenario, VMs/VNFs of both mission critical TAs and emerging fog applications are optimally placed in the fog network. All emerging fog apps considered are moderately sensitive to end-to-end delay in the network and the network has a trivial queuing delay penalty i.e., $\delta \ll 1$. The scenario is evaluated when TS architecture is adopted and when the DS architecture is adopted in fog computing servers.

6.4.2.1 Placement under Traditional Server Architecture

The illustration in Figure 6.8 shows the optimal placement of emerging fog applications when TSs are deployed in the fog network. A single instance of each emerging fog app is provisioned in the fog network. The instance

provisioned for each emerging fog app satisfies the computing capacity requested by all geo-distributed users of that app. These provisioned instances are strategically placed in the fog network to minimise both TNPC and TFPC. All mission critical TAs are also placed in their respective associated nodes to minimise the high cost of rejecting them as defined in the objective. However, network power consumption is dominant because a small fog network with a limited number of fog computing nodes, users and applications is considered. Moreover, fixed regular traffic in the network topology accounts for a significant portion of the TNPC.

Scenario	Low delay penalty				High delay penalty		Low delay penalty with delay sensitive fog app "X"			
Server Architecture	TS		DS		TS	DS	TS		DS	
Network Node (type)	MILP	HEEDAP	MILP	HEEDAP	MILP		MILP	HEEDAP	MILP	HEEDAP
1 (CO)	◆★	◆★			◆★		◆	◆		
2 (CO)					◆	◆●				✚
3 (CO)	●	●			●		●	●		
4 (CO)			◆★	◆★		◆★				◆
5 (EO)										
6 (Radio CS)	✚	✚	●		★	●	★✚	★✚	★✚◆	★
7 (Home)										
8 (Home)										
9 (EO)										
10 (Radio CS)			★		★	★	★	★	★	★
11 (Home)										
12 (Home)										
13 (EO)										
14 (Radio CS)				●	◆★	✚	★	★	★	★
15 (Home)										
16 (Home)										
17 (EO)										
18 (Radio CS)			✚	✚	◆★✚	●★	★	★	★●	★●
19 (Home)										
20 (Home)										
● - Fog app U ✚ - Fog app U ◆ - Fog app W ★ - Fog app X										

Figure 6.8: Energy efficient placement of emerging fog apps in a fog network.

Figure 6.8 shows that there is high preference for metro COs when the selection of optimal locations for emerging fog apps is made. This is often the situation when surplus compute capacity is available in metro COs. Relative to other network nodes, COs are centrally located, closer to geo-distributed

users and closer to the metro gateway to the cloud. Hence, placement of fog apps in COs reduces the number of hops travelled by the fog traffic and therefore reduces TNPC. As illustrated in Figure 6.8, instances of three emerging fog apps are placed in metro COs. The power consumption incurred by hosting a given emerging fog app in inactive resource components is the same across all network nodes. The is because homogenous servers are adopted across the distributed fog computing nodes. Likewise, power consumption incurred by hosting a given emerging fog app in the idle resource capacity (IRC) of active resource components is also equal across all network nodes. Therefore, multiple candidate metro COs which lead to the same increment of TFPC may exist for a given emerging fog app.

Such ties are broken by selecting the metro CO which enables lower total (i.e., fog computing plus network) power consumption. Network node 1, which is also the metro gateway node to the core network in the network topology, wins such tie breaks. This is because the placement of emerging fog apps close to the metro gateway helps to reduce the number of hops traversed in the network topology. Consequently the TNPC is also reduced. Emerging fog apps “W” and “X” are placed in network node 1 as shown in Figure 6.8. However, when resource capacity in network node 1 is limited, other candidate metro COs with adequate compute capacity are considered. Consequently, network node 3 is selected to host emerging fog app “U” as shown in Figure 6.8.

In the absence of surplus fog compute capacity in metro COs, fog sites in the access network must be selected to support emerging fog apps. Candidate fog sites in the access network must have adequate surplus computing capacity. However, multiple candidate access nodes may also present equal compute energy efficiency to host a given emerging fog application. This is because of the homogeneous power profile of fog computing nodes across the distributed fog network. Hence, network energy efficiency is used as a decision metric to select the optimal network node in such scenarios. For example, emerging fog app “V” is placed in network node 6, an access node as illustrated in Figure 6.8. Fog app “V” requires a dedicated server because of the intensive nature of its memory demand. Node 6 is selected over other network nodes (10, 14 and 18) due to its proximity to the metro gateway node. This choice promotes lower total network traffic because the number of hops traversed by cloud bound traffic is reduced. It is important to note that unused servers are also present in network nodes 10, 14 and 18 as illustrated in Figure 6.4.

6.4.2.2 Placement under Disaggregated Server Architecture

Replacing TSs with DSs in the fog network leads to changes in the optimal placement of emerging fog applications as shown in Figure 6.8. Improved consolidation of both traditional and emerging fog apps enabled by the adoption of DSs in the distributed fog network is responsible for the revised placement observed. Consequently, Figure 6.9 shows corresponding increases in the average utilisation of active resources components in the fog network that was achieved because of the revision in server architecture. When the DS architecture is deployed in the fog network, a primary instance of each emerging fog app is provisioned in the network node that leads to optimal energy efficiency. Additionally, secondary instances of an emerging fog app may be created. The creation of secondary instances should lead to marginal rise in TFPC while enabling significant drop in the TNPC. Reductions in TNPC is achieved because the creation of secondary instance(s) enable reductions in the number of hops traversed. This justifies the creation of two instances of emerging fog app “X”. The instance in network node 18 is responsible for distributed users of the application in network nodes 6, 7, 8, 14, 15, 16, 18, 19 and 20. A second instance of emerging fog app “X” in network node 10 is responsible for distributed users in network nodes 10, 11, and 12. Hence, the number of hops between instances of app “X” and their distributed users is minimised. It is important to note that the instance in node 10 is provisioned using IRC of active resource components. Hence, minimal power is consumed to create the additional instance of fog app “X”. This instance enables good reductions in TNPC.

Relative to results obtained when TSs are deployed in the fog network, the revisions in fog apps placement observed when DSs are deployed is responsible for about 18% fall in TFPC as shown in Figure 6.10. Reduction in the TCPC is responsible for over 90% of the fall seen in TFPC. This is because power consumption of CPU component is significantly higher than that of memory and storage components. Disaggregation enables significant improvements in CPU utilisation efficiency (as shown in Figure 6.9). Improved consolidation of CPU demands of mission critical traditional apps and emerging fog apps in each fog computing site is responsible for this. Hence, the number of active CPU component reduced when DS are deployed to replace TS in the fog network as shown in Figure 6.11. The same is also true for storage components and their corresponding utilisation efficiency. However, the 33% drop in the number of storage components observed in Figure 6.11 does not lead to significant fall in the TFPC. This is because

storage components have a lower peak power consumption relative to CPU and memory components as illustrated in Table 6.2. Figure 6.11 only shows a marginal drop in the number of active memory components. This is because several considered applications have high memory demand relative to the capacity of the homogenous memory components as given in Table 6.3. Hence, a significant improvement in active memory utilisation could not be realised as shown in Figure 6.9.

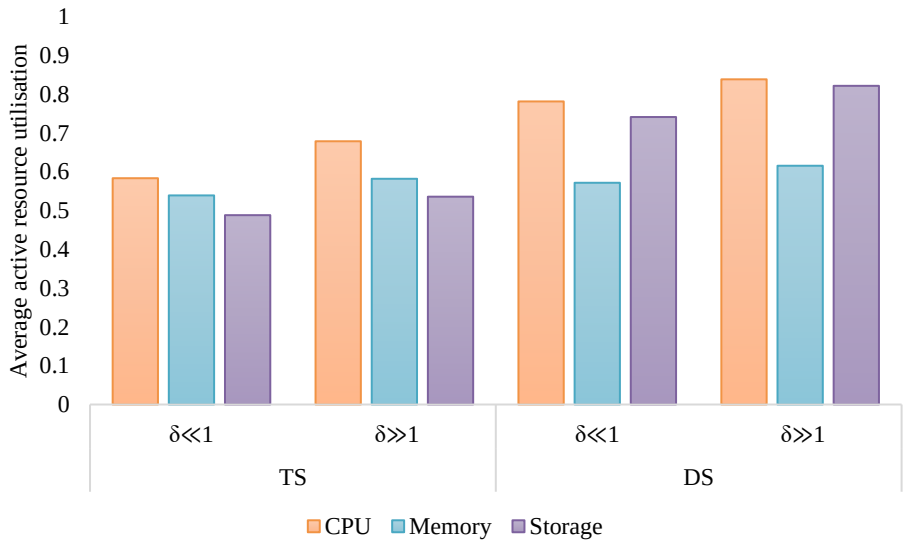


Figure 6.9: Average utilisation of active components across fog sites

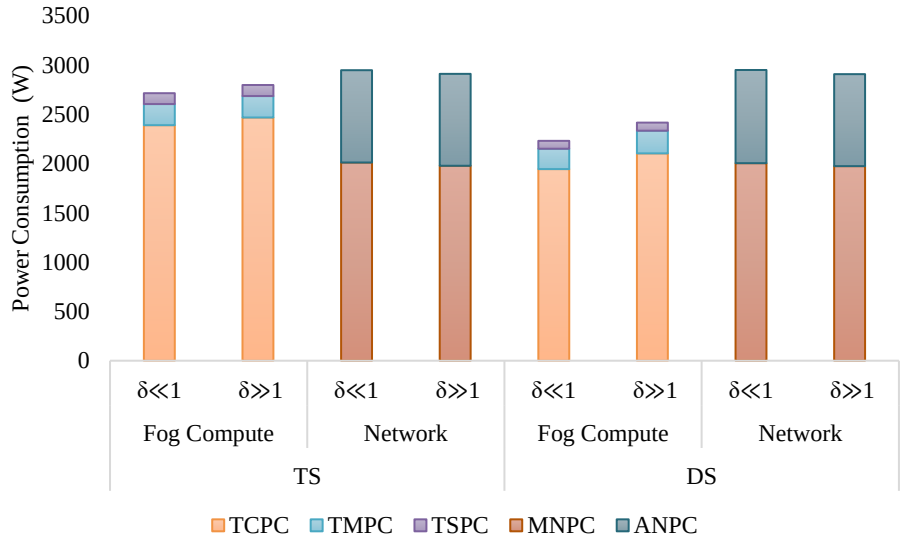


Figure 6.10: Power consumption under EE placement scenarios.

Figure 6.10 shows a marginal increase in the TNPC after TSs are replaced with DSs in the fog network. This marginal rise is bceause of increased hop count between the instance of some emerging fog apps and their users. Furthermore, the hop count between instance of emerging fog app

“W” and “X” and the metro gateway node also increased after a change in server architecture. Hence, network traffic traverses more network equipment when DSs are deployed in the fog network. Figure 6.12 gives the average and maximum round trip time (RTT) between distributed user of each emerging fog app and the provisioned instances of the app. Relative to the deployment of TS architecture in the fog network, both average and maximum RTT increased when DS architecture is adopted in the fog network. Users of emerging fog app “U” experience relatively higher delay as shown in Figure 6.12 compared to other emerging fog apps. Although the placement of fog app “U” in network node 6 enables optimal energy efficiency in the fog network, this choice also leads to high congestion on the link that connects network node 6 to node 1. Consequently, users of emerging fog apps “V”, “W” and “X”, which are located in node 6, experience the corresponding maximum delay illustrated in Figure 6.12. Since, the moderate delay thresholds of all emerging fog apps under this scenario are satisfied, such performance is acceptable. However, the performance obtained under both TS and DS architectures violates the delay threshold (i.e., sub-20ms) for delay sensitive fog apps. Hence, another subsection considers a scenario where requests for a delay sensitive emerging fog app are present in the fog network.

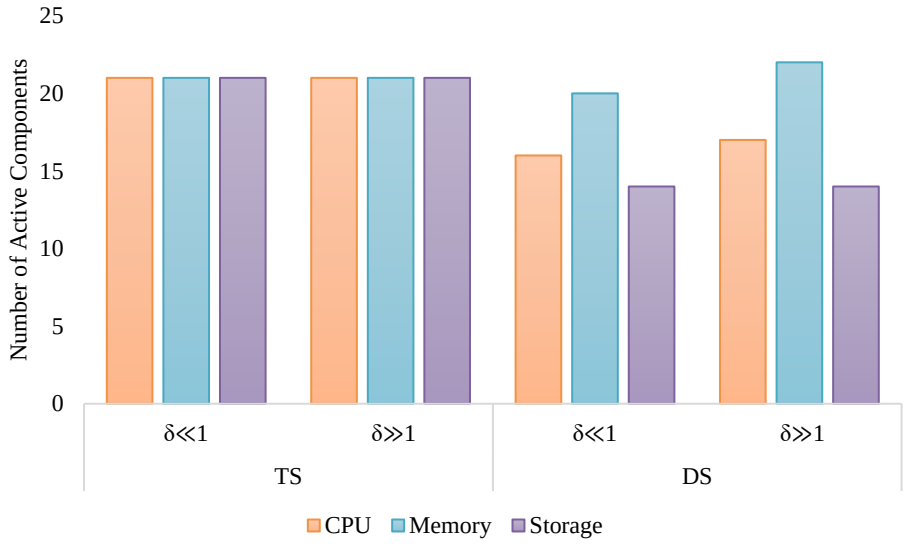


Figure 6.11: Number of active components across fog sites

Figure 6.12: Round trip delay from app instance to users under EE placement scenarios.

6.4.3 Energy Efficient Placement under High Delay Penalty

In this subsection, $\delta \gg 1$; hence, network delay penalty is high. This represents a network where the operator desires minimal impact of emerging fog apps on regular traffic.

6.4.3.1 Placement under Traditional Server Architecture

Under this scenario, multiple instances of some emerging fog apps are created when the TS architecture is adopted as shown in Figure 6.8. This reduces the number of hops between instances of replicated fog apps and their users. Consequently, the total volume of traffic traversing the network topology is reduced and the delay experienced on each link of the network topology is also minimised. Relative to result obtained under low delay penalty ($\delta \ll 1$), the average and maximum RTT between the instance of fog apps and their distributed user falls when $\delta \gg 1$ as shown in Figure 6.12.

However, to ensure a balanced trade-off between minimising TFPC and the total approximated delay, only applications (app “W” and app “X”) with low-medium resource demand intensity are replicated as shown in Figure 6.8. The primary instances for both apps are placed in centrally located fog sites (i.e., COs). These primary instances are responsible for users in directly attached access network nodes and also for most distributed users in the network topology. On the other hand, additional instances of emerging fog app “W” and app “X” are provisioned in some radio CSs. This strategy reduces the traffic associated with densely populated user clusters attached to each radio CS in the network. This consequently reduces network congestion. Because

the app “W” and app “X” have small compute footprint, they are easily provisioned with IRC of active TSs. Hence, minimal increase in TFPC is incurred compared to the $\delta \ll 1$ scenario as shown in Figure 6.10. Furthermore, relative to results obtained when $\delta \ll 1$, additional compute components are not activated to provision the additional instances of app “W” and “X” as shown in Figure 6.11. However, replication of fog app instances leads to the increase observed in the utilisation of active resource components in the fog network as shown in Figure 6.9. Relative to the low delay penalty scenario where TSs are deployed, there is a 3% rise in the TFPC when the high delay penalty scenario is considered under a similar setup. On the other hand, the TNPC consumption fell by 2% as shown in Figure 6.10.

6.4.3.2 Placement under Disaggregated Server Architecture

A similar trend is observed when DSs are deployed in the fog network. Replicas of certain emerging fog apps are made, as shown in Figure 6.8. This strategy enables reductions in the volume of traffic traversing the network topology. It also reduces increased network congestions that would have occurred due to the creation of a single instance of each fog app. Consequently, distributed users of fog apps experienced lower average and maximum RTT to assigned fog instances as shown in Figure 6.12. Under the high delay penalty scenario, three instances of app “U” and app “X” are created. Only two instances of fog app “W” are created under the high delay penalty scenario. High CPU and memory demands of emerging fog app “V” prevents the replication of the fog app. This helps to avoid significant increase in TFPC. As observed when TSs are deployed in the network, radio CSs are often used to host instances of emerging fog apps. This helps to ensure that the total volume of traffic in the network topology is minimised. This is because of the high user density associated with radio CSs in the system setup. For example, fog app “V” is placed in network node 14 because the node does not require the activation of an additional CPU component to support the CPU intensive demands of emerging fog app “V”.

Replication of app “U” and app ”W” can significantly reduce the total volume of traffic and congestion experienced in the network. This is because these emerging fog apps have relatively higher data rate per user than other emerging fog apps as illustrated in Table 6.3. Additionally, app “X” has non-intensive CPU and memory demands; therefore, the replication of app “X” does not lead to significant increase in the TFPC but can lead to additional reduction in the network traffic.

Under a similar setup as the low delay penalty scenario, the creation of replicas of some emerging fog apps when the DSs are deployed under the high delay scenario leads to about 8% rise in the TFPC as seen in Figure 6.10. A comparison of TNPC under both scenarios shows about 1% decrease due to reduced traffic on the network topology as illustrated in Figure 6.10. Relative to the low delay penalty scenario where DSs are used, the replication of some emerging fog apps (“U”, “W” and “X”) under a similar setup in the high delay penalty scenario increased the number of active compute components as shown in Figure 6.11. A similar comparison also shows that the average active utilisation of compute components increased in the fog network as shown in Figure 6.9.

Comparison of TS and DS architectures under the high delay penalty scenario expectedly shows that the adoption of the DSs enabled notable (about 14%) reduction in TFPC as shown in Figure 6.10. This is because server resource components are independently and proportionally utilised. A marginal fall in TNPC is also observed as a result of the revised server architecture. This is because the DS architecture encouraged the creation of more distributed replicas of most emerging fog apps relative to when the TS architecture is adopted. Compared to the placement obtained under TS architecture where replication of emerging fog app “U” is discouraged because of the app’s high compute footprint, replicas of emerging fog app “U” are created when the DS architecture is deployed. Note that app “U” has moderate CPU demand and high memory demand. Hence, proportional usage of resource components when DS architecture is employed promotes the independent activation of new memory components to support replicas of app “U”. The moderate CPU demands of the fog app U’s replicas are aggregated with the CPU demand of other applications into active CPU components. However, replication of app “V” is still discouraged because of its high CPU and memory demands. Therefore, proportional usage of resource components does not enable sufficient benefits to promote replication of an emerging fog app that is CPU and memory intensive. Relative to the deployment of TS under the high delay penalty scenario, Figure 6.12 shows that the average and the maximum round trip time are higher for some emerging fog apps when DSs are employed. Thus, the number of hops between users of such emerging fog apps and the instances of the app that is assigned to them increased. This is because more network nodes are traversed.

6.4.4 Energy Efficient Placement of Delay Sensitive Fog App

A network with trivial queuing delay penalty ($\delta \ll 1$) is adopted under this scenario. In contrast with the previous scenario, emerging fog app "X" is sensitive to end-to-end delay in the network (i.e., sub-20ms delay requirement) under this scenario. Other emerging fog apps remain moderately sensitive to end-to-end delay in the network as in the previous subsections. Under this scenario, EE placement of both traditional and emerging fog apps is also evaluated when both TS and DS architectures are deployed in the fog computing nodes placed in the fog network.

Multiple instances of the delay sensitive emerging fog app are provisioned at all radio CS in the network topology as shown in Figure 6.8. This observation is common for both server architectures adopted in fog nodes. This strategy ensures that the delay threshold of the fog app "X" is satisfied for users that are directly attached to a radio CS. On the other hand, users of emerging fog app "X" which do not have direct access to a radio CS are rejected. Hence, local computation capacity, which can lead to higher CAPEX and OPEX, is required to support such users. It is expected that a similar placement strategy will be implement if users of emerging fog apps are associated with enterprise office. Other emerging fog apps, which are moderately sensitive to delay, are placed to achieve optimal energy efficiency in the fog network as reported in Section 6.4.2. Figure 6.13 shows that the TNPC increases marginally when DS architecture is deployed in fog sites to replace TS architecture as observed previously.

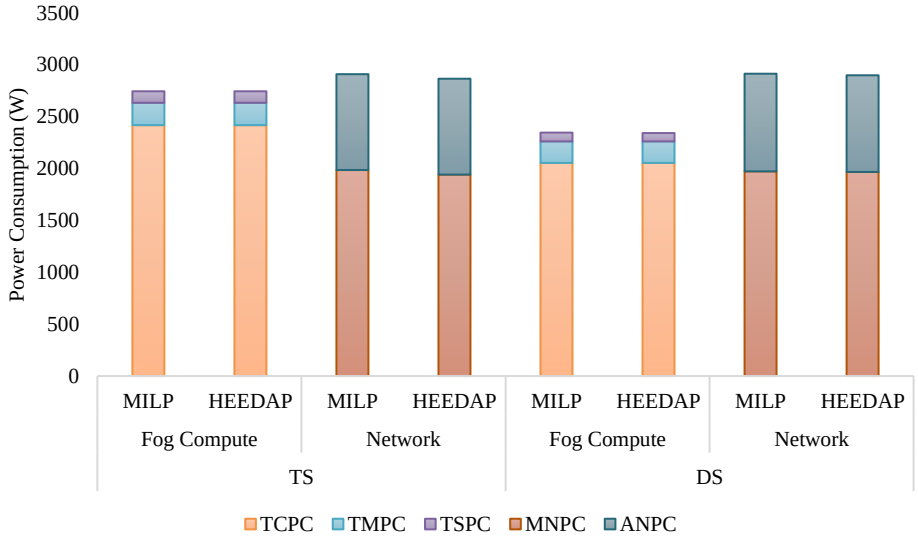


Figure 6.13: Power consumption under energy efficient placement of a delay sensitive fog app scenario

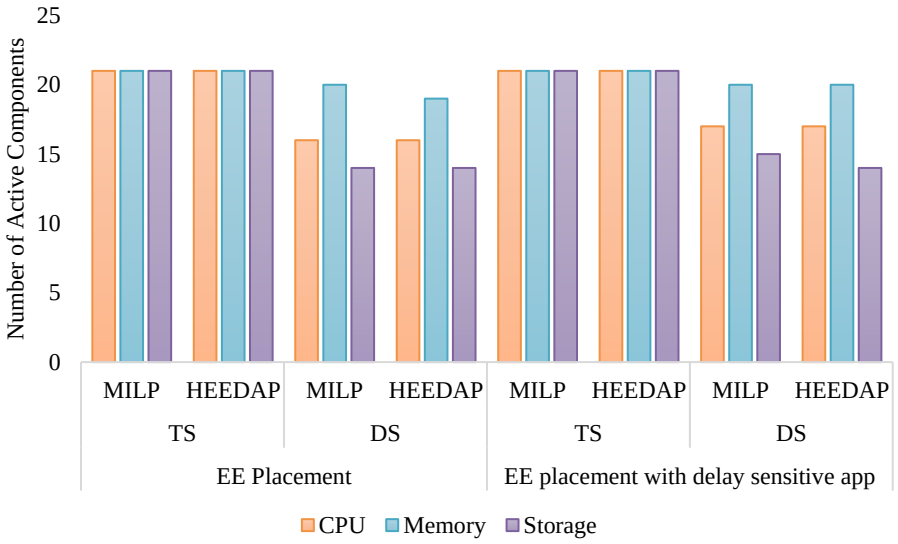


Figure 6.14: Number of active components across fog sites

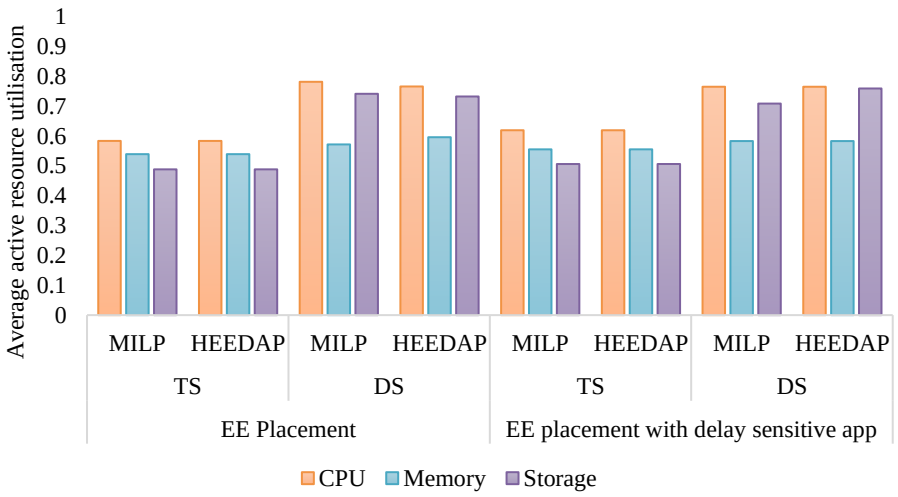


Figure 6.15: Average utilisation of active components across fog sites

Furthermore, the TFPC and the number of active components is lower when DSs are used to replace TSs in the fog network as shown in Figure 6.13 and Figure 6.14 respectively. The adoption of DS architecture in the fog network increases the amount of spare capacity available to support both highly sensitivity and moderately sensitive emerging fog apps at the network edge without additional CAPEX. This is because the adoption of DSs reduces the number of active resource components while provisioning the same number of emerging fog apps as when TS architecture is adopted. As observed in Section 6.4.2, Figure 6.16 shows that the average and maximum delay experienced by distributed users of moderately delay-sensitive emerging fog apps increases when DS architecture is adopted. For instance, the illustration in Figure 6.8 shows that each instance of moderately sensitive emerging fog apps created are provisioned in radio CSs. These radio CSs are

far from most of the distributed users of each fog app. However, the performance of such applications does not degrade because they have greater delay-tolerance. It is important to note that the delay experienced by served users of the delay sensitive apps is considered to be extremely low and insignificant as expected of today's 5G mobile networks and future 6G mobile networks.

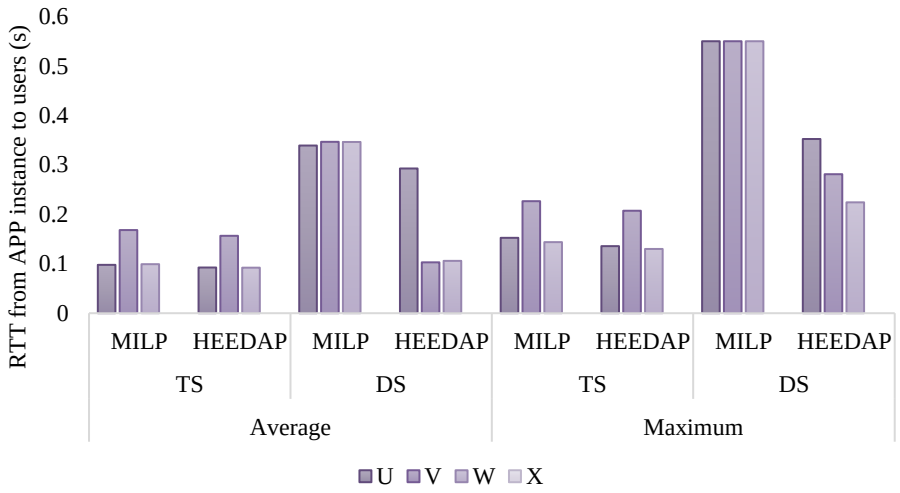


Figure 6.16: Round trip delay experienced by users of emerging fog app.

6.5 Heuristic for Energy Efficient and Delay Aware Placement of Fog Applications

The results obtained in the previous section reveal the ability of the formulated MILP model to perform best-effort placement of mission critical and emerging fog applications in a fog network. This is done in an energy efficient manner without violating application specific delay requirements. However, due to the complexity of the MILP model, the performance of the MILP model is only evaluated for a small fog network. This complexity increases exponentially as the size of the fog network is scaled-up. Hence, massive computing capacity and time (days) are required to solve the MILP model for practical deployment in large fog networks. This is neither energy efficient nor practical because control and orchestration mechanisms of fog networks must make placement decisions in near real-time to ensure optimal user experience and fog applications' performance. Therefore, a fast heuristic that mimics the insights obtained from the MILP model is a more ideal solution for fog networks.

To obtain results that approach those of the MILP model, a policy that leverages a centralised orchestration and management framework for a network of distributed fog computing nodes is required. Such centralised

control is an essential tool that can enable the efficiencies observed during the analysis of results from the solved MILP model. Hence, a heuristic is proposed. The heuristic depends on centralised control of distributed fog nodes to achieve high efficiency in a fog network. It is also leverages global knowledge derived from control information exchange to achieve high efficiency in a fog network. The heuristic is a real-time algorithm that optimally provisions both mission critical traditional applications and (delay sensitive) emerging fog applications in a fog network when possible.

Given a set of input fog apps (i.e., mission critical traditional apps and emerging fog apps), the algorithm attempts to provision instance(s) of these applications in a fog network in an energy efficient manner while considering the delay requirements of each application. Applications that cannot be provisioned are rejected and users whose delay requirements are not satisfied by provisioned instance(s) are also rejected. The algorithm supports the use of TS and DS architectures in the fog network. Other inputs to the algorithm include the user distribution and delay requirement of each emerging fog app; the distribution of fog compute nodes in the network topology; the features and characteristics of each resource components at each fog site; and the load and propagation delay on each network link.

6.5.1 HEEDAP Algorithm Description

A high-level description of the heuristic for energy efficient and delay aware placement (HEEDAP) of applications in fog networks is illustrated in Figure 6.17. At inception, the HEEDAP algorithm processes the set of input mission critical TAs in each network node. The list of TAs in each network node is sorted in descending order of CPU demand intensity. If a tie occurs, memory demand intensity is initially adopted to break the tie followed by storage demand intensity. The output of this process is the “local job list” (LJ-list) created at each network node.

Secondly, the HEEDAP algorithm processes the set of input emerging fog apps to the fog network in two stages. The initial stage identifies fog apps that are highly sensitive to network delay. These fog apps form the secondary job list at each fog computing site. A delay sensitive fog app is placed in secondary job list of a fog site if some users of that app are directly connected to the corresponding fog site via wireless media. This secondary job list created at each fog computing site is called the “pseudo-local job list” (PLJ-list). Emerging fog apps that are classified as delay sensitive are subsequently ejected from the list of input emerging fog apps. The PLJ-list at each network node is arranged in descending order of resource demand intensity as

described for the LJ-list. In the second stage, the HEEDAP algorithm sorts the list of input (moderately sensitive) emerging fog apps in descending order of resource intensity to create the “real fog job list” (RFJ-list). The RFJ-list also (implicitly) holds information about the user distribution. User distribution information is represented by the number of users that made a request for each emerging fog app at each network node. After input apps processing, HEEDAP creates a temporary copy of the RFJ-list. This temporary copy is the “pseudo fog job list” (PFJ-list) which is refreshed after each complete iteration of the algorithm. It is important to note that an iteration of the HEEDAP algorithm is complete when the PFJ-list of that iteration is empty. Furthermore, a union of LJ-list and PLJ-list at each network node and the RFJ-list form the list of applications in the fog network.

Whilst the list of applications in the fog network is not empty (this is the first check of the HEEDAP algorithm), at each network node with compute capacity a “query app” is selected. The mission critical traditional fog app at the top of the LJ-list in each network node is selected as the query app. The query app at each network node is placed energy-efficiently; new resource components may be activated to support the query app as required. If the query app could not be placed, it is rejected and removed from the LJ-list. The state of all compute resource components in each network node is recorded and stored by the central orchestrator.

Thereafter, at each network node, a candidate app in the LJ-list is identified to maximise the utilisation of the IRC of active components in this network node where possible. If a candidate app is not found in the LJ-list, the PLJ-list is checked for a candidate app. The search for a candidate local or pseudo-local app in each network node gives higher priority to maximum utilisation of the IRC of active CPU components. This is because CPU components consume more power than memory and storage components as illustrated in Table 6.2. When the DS architecture is adopted, inactive memory and storage components within the same network node may be used to complement available IRC of CPU component. A similar approach may be adopted when the TS architecture is deployed and a single compute node has multiple intrinsic CPU, memory, and storage components; otherwise, the resource locality constraint of TS architecture is enforced.

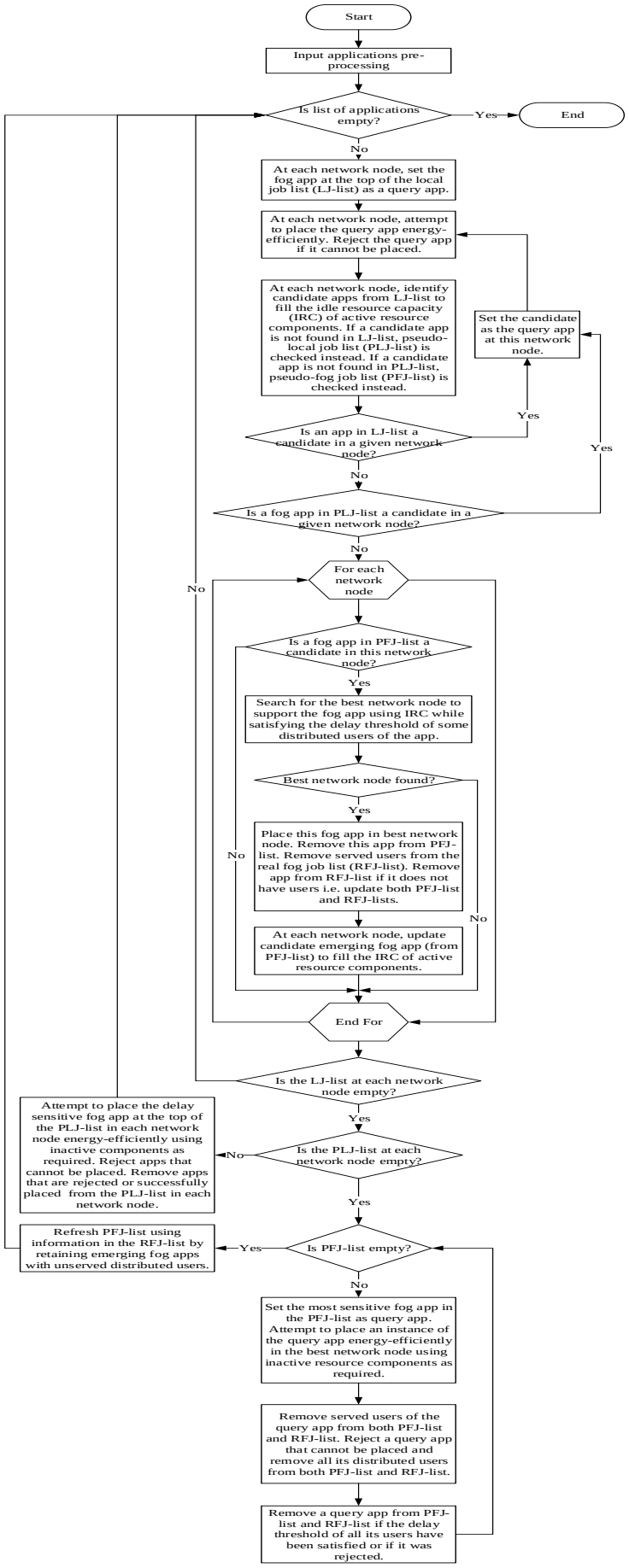


Figure 6.17: Flow chart of HEEDAP algorithm.

If a candidate app is not found in the LJ-list or PLJ-list of a network node, the PFJ-list is searched to identify a moderately sensitive emerging fog app that can maximise utilisation of the IRC of active components of such network node. This search gives priority to the emerging fog app with more stringent delay requirement if the node is within the round-trip delay threshold of one or more unserved users of that fog app. The search conducted at each network node provides control data that supports the placement of emerging fog apps in the PFJ-list in subsequent steps of the HEEDAP algorithm. Such control information provides the global knowledge required by the algorithm.

In each network node, if a candidate app is found in the LJ-list, the app is selected as the new query app and placed energy-efficiently. Hence, the algorithm gives higher priority to mission critical traditional fog apps of the fog computing infrastructure provider. Otherwise, if a candidate app is found in the PLJ-list, the app is also selected as the new query app and is placed energy-efficiently. Relative to moderately sensitive emerging fog apps, this gives greater priority to emerging fog apps that have greater delay sensitivity. On the other hand, if a candidate app is not found in LJ-list or PLJ-list in a given network node and one is found in the PFJ-list, further checks are required. The algorithm checks if other network nodes can also host this candidate emerging fog app using IRC before the best network node for the fog app is selected. Thus, global knowledge of the central orchestrator must be consulted before the best network node is selected from the list of all contending network nodes. These contending network nodes are nodes that can provision the candidate (moderately sensitive) emerging fog app using IRC.

The best network node for a given moderately emerging fog app in the PFJ-List is the network node that leads to the smallest increase in TFPC after the placement of the candidate emerging fog app i.e., the most energy efficient network node. However, ties may occur when energy efficiency is used as the decision metric. Hence, the CO that is closest to the metro gateway node is given the highest priority when a tie occurs. This ensures that the (number of hops traversed or) traffic in the network is minimised as observed in the analysis of the MILP model results. Furthermore, if the list of contending network nodes for the emerging fog app comprise of only access network nodes, the network node with the highest user density is given higher priority. Otherwise, if some contending network nodes have the same user density for the emerging fog app, then lower delay to the metro gateway node is used as the decision metric to select the best network node.

Once the best network node for a given emerging fog app in the PFJ-list is found, an instance of the app is provisioned in that network node. All users of the app that can be served by this new instance (without violating delay requirements and network capacity constraints) are removed from the RFJ-list. Furthermore, the fog app and its served users are also removed from the PFJ-list of the present iteration. Additionally, the provisioned emerging fog app is replaced as a candidate emerging fog app at all corresponding fog computing sites where it was previously a candidate. A new candidate fog app from the PFJ-list is elected to maximise the utilisation of the IRC of active CPU component at such network nodes. This approach ensures fair placement for each emerging fog app in the fog network. This placement strategy is repeated at all network nodes to place all elected candidate emerging fog apps in the active iteration.

After exhausting all opportunities to use the IRC to provision an instance of each candidate emerging fog app in the PFJ-list in the active iteration, the size of the LJ-list is checked. If the LJ-list is not empty, the HEEDAP algorithm repeats all procedures described above. Hence, the algorithm attempts to provision all mission critical traditional apps before considering emerging fog apps for placement. On the other hand, if the LJ-list is empty, another check is made to confirm that the PLJ-list in each network node is empty. If the PLJ-list in a network node is not empty, the HEEDAP algorithm attempts to provision the fog apps at the top of the PLJ-list in each network node. Such attempts may activate inactive resource components as required since the LJ-list in the network node is now empty. Hence, after mission critical TAs, emerging fog apps with higher sensitivity to delay have the highest priority. Emerging fog apps in the PLJ-list of a network node that are placed successfully are removed from the PLJ-list of that network node. Fog apps that are rejected are also removed from the PLJ-list of the corresponding network node. The HEEDAP algorithm thereafter repeats all procedures described above to place all moderately-sensitive emerging fog apps with available IRC of active resource components. It is important to note that if both LJ-list and PLJ-list of a network node are empty, only fog apps in the PFJ-list of the active iteration will be considered for energy efficient placement. Energy efficient placement uses IRC as described in earlier steps of the algorithm.

On the other hand, if the PLJ-list is empty, a new check is made to confirm if the PFJ-list of the active iteration of the fog network is empty or not. If the PFJ-list is not empty, the emerging fog app with the highest delay sensitivity in the PFJ-list is set as query app and energy-efficient placement

of the query app is attempted. Inactive resource components may be activated as required to provision an instance of this query app. If a query app cannot be provisioned, the app is deleted from the PFJ-list and RFJ-list along with the information about all un-served distributed users of that app. Otherwise, if an instance of the query app was successfully provisioned, users of the query app whose delay threshold has been satisfied by the new instance are removed from both PFJ-list and RFJ-list. To ensure fairness when placing emerging fog apps, the provisioned query app is also removed from the PFJ-list of the active iteration. However, the details of unserved distributed users are retailed in the RFJ-list. Thereafter, the HEEDAP algorithm repeats all previous steps in this paragraph to provision one instance of each moderately sensitive emerging fog apps in the PFJ-list until the list is emptied.

An empty PFJ-list implies that an active iteration of the HEEDAP algorithm has been completed and that a single instance of each emerging fog app has been provisioned. Recall that the HEEDAP algorithm uses the RFJ-list to maintain global knowledge of users of some emerging fog apps whose delay requirement remain unfulfilled. This knowledge is used to refresh PFJ-list. The refreshed PFJ-list comprise of all emerging fog apps with one or more unsatisfied users. The HEEDAP algorithm subsequently returns to the first check at the top of the algorithm to begin a new iteration since this check will be negative. However, if the delay requirements of all users of all emerging fog apps has been satisfied or all requests for the emerging fog apps has been rejected the first check of the algorithm will be positive and the HEEDAP algorithm stops. Note that user request for an emerging fog app could have been rejected because delay threshold could not be satisfied.

The HEEDAP algorithm calculates delay by considering the sum of the delay experienced on a link and link’s propagation delay as the delay cost on each link in the network topology as given in the MILP model. The path with the smallest total delay is always selected as the shortest path between two nodes. It is assumed that the information of the network topology such as propagation delay and historical traffic (load) on each network link is available as input to the HEEDAP algorithm. Given this information, Dijkstra's shortest path algorithm is used determine the shortest path between two network nodes using total (propagation plus congestion) delay as the cost metric.

In the HEEDAP algorithm, resource locality constraint distinguishes a fog computing site with DS architecture from a fog computing site with TS architecture. To reduce the complexity of control and orchestration required for the algorithm in a large fog network deployment, big fog networks can be

sub-divided into multiple small units. The algorithms can be deployed in a stand-alone mode in each small unit. Criteria for deciding the division thresholds for big fog networks include delay, number of network nodes and fog application user distribution. It is also important to note that an emerging fog app is a candidate app in a network node under the following condition. That is, if and only if network capacity exists on a selected shortest path that satisfies the delay threshold of some users of that fog app after the placement of that fog app into the node. Otherwise, the fog app is not an acceptable candidate for that network node. Similarly, the users of provisioned fog app at a given network node are removed from the RFJ-list when this condition is satisfied. That is, if and only if the delay threshold of such users are satisfied within specified capacity constraint of links on the selected shortest path between users of that app and the network node where the instance has been provisioned.

6.5.2 HEEDAP Performance Evaluation

To evaluate the performance of the HEEDAP algorithm, two evaluation scenarios (i.e. EE and EE placement of delay sensitive fog app) studied with the MILP model in previous section are considered. The results obtained when the algorithm is deployed in these scenarios are compared with those obtained by solving the MILP model. Similar computing and network metrics given in the previous section are also adopted. It is assumed that fog network providers have SLAs with partners (network service providers) that guarantees shortest path delay for emerging fog apps in the network topology. This is because the evaluation of fog networks considered a network infrastructure that is shared with other services,.

6.5.2.1 Energy Efficient Placement

In the absence of fog applications which are sensitive to delay under the EE placement scenario, Figure 6.18 shows that the HEEDAP algorithm achieves comparable results as those reported when the MILP model is solved. As shown in Figure 6.18, Figure 6.14 and Figure 6.15, the HEEDAP algorithm achieves the same TFPC, number of active resource components and average active resource utilisation as the MILP model when the TS architecture is deployed in the fog network. This demonstrates the efficacy of the HEEDAP algorithm at mimicking the compute energy efficiency achieved by the MILP model in a similar system setup that adopts homogenous resource components across the fog network. A single instance of each emerging fog app is provisioned in the fog network as reported when the MILP model was solved. As shown in Figure 6.8. The placement of the instance

created for each emerging fog app obtained via the HEEDAP algorithm is also an exact match with those obtained by solving the MILP model when TS architecture is deployed in fog nodes. Consequently, the average and maximum RTT from app instance to users obtained by HEEDAP are also comparable to those obtained by solving the MILP model when TS architecture is adopted in fog network nodes as shown in Figure 6.16. The TNPC obtained using the HEEDAP algorithm is marginally (about 2%) lower than that of the MILP model as shown in Figure 6.18. Disparity in path selection made for cloud bound traffic is responsible for this difference. The MILP minimises overall congestion in the network topology by distributing such traffic as necessary. This leads to higher network power consumption. On the other hand, HEEDAP always selects the shortest path this approach minimises network power consumption.

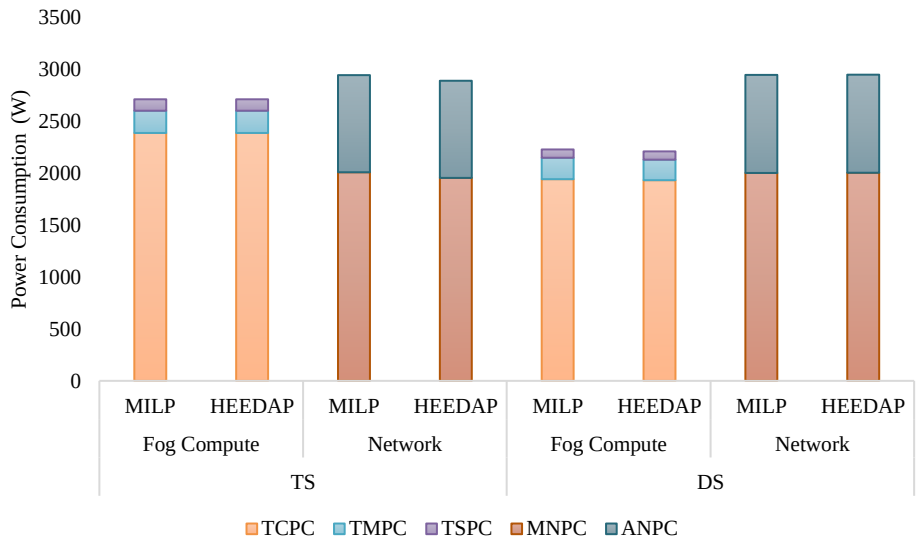


Figure 6.18: Power consumption under energy efficient placement scenario

Similar trends are also observed when the HEEDAP algorithm is deployed to perform EE placement of moderately sensitive emerging fog apps in a fog network that adopts DS architecture. The TFPC obtained by HEEDAP algorithm is almost equal to that obtained by solving the MILP model. The TFPC of the HEEDAP algorithm is marginally (1%) lower. This is because only a single instance of emerging fog app “X” is provisioned when HEEDAP algorithm is deployed as shown Figure 6.8. Two instances of emerging fog app “X” are created when the MILP model was solved. Consequently, relative to the TNPC obtained by solving the MILP model, the TNPC obtained by HEEDAP algorithm is marginally higher because the total volume of traffic in the network is higher. Furthermore, the placement of emerging fog apps obtained by the HEEDAP algorithm as shown in Figure 6.8 is largely

comparable to those obtained by solving the MILP model. However, compared to results obtained by solving the MILP model, the placement of emerging fog app “U” as obtained by the HEEDAP algorithm is different since the app is placed in node 14. This revised placement is responsible for the fall in the average and maximum RTT experienced by the distributed users of the app as shown in Figure 6.16. This is because node 14 is farther away from node 1 which is also the metro gateway node to the cloud. Hence, the congestion on the paths to node 14 is lower.

6.5.2.2 Energy Efficient Placement of Delay Sensitive Fog App

In the presence of a delay sensitive emerging fog application i.e., app X, the HEEDAP algorithm is able to mimic the performance of the MILP when TS or DS architecture is adopted in the fog network. The resulting placement of emerging fog apps, as depleted in Figure 6.8, also shows this. By pre-processing input application in the initial steps of the HEEDAP algorithm, placement or rejection of delay sensitive fog apps is simplified. This simplification is irrespective of the presence or absence of in situ computing capacity at the source of user request for delay sensitive fog apps. Therefore, the HEEDAP algorithm effectively mimicked the MILP model by provisioning some instances of delay sensitive emerging fog apps at radio CSs to serves users at such location. Users of delay sensitive emerging fog apps located at network nodes without local computing capacity are rejected by the fog network as discussed previously.

When TS architecture is deployed in the fog network, the resulting placement of emerging fog apps by the HEEDAP algorithm is an exact replica of the placement obtained by solving the MILP model. Consequently, the same TFPC is achieved by both the MILP model and HEEDAP algorithm under the corresponding server architecture as shown in Figure 6.13. As shown in Figure 6.14, the HEEDAP algorithm obtained the same number of active resource component as those obtained by solving the MILP model. Likewise, HEEDAP also replicates the average utilisation of active components across fog computing sites obtained by solving the MILP model as show in Figure 6.15. The TNPC obtained by the HEEDAP algorithm is also comparable to the same value obtained by solving the MILP model under a similar scenario. Similarly, the average and maximum RTT to distributed users of emerging fog apps obtained by the MILP model is also comparable to those obtained by the solving the MILP model as shown in Figure 6.16.

However, when the DS architecture is employed in the fog network, the resulting placement of emerging fog apps by the HEEDAP algorithm is not an

exact replica of the placement obtained by solving the MILP. The TFPC obtained with the HEEDAP algorithms is about 2% higher than the TFPC obtained by solving the MILP model under this scenario as shown in Figure 6.13. The adoption of homogeneous compute resource across the fog network is responsible for the comparable TFPC obtained. Therefore, the number of active resource components and the average utilisation of these active components across fog computing sites as obtained by the HEEDAP algorithm is comparable to those obtained by solving the MILP model. Difference in the placement of emerging fog apps is responsible for the changes in the average and maximum RTT experienced by users as shown in Figure 6.16. Apps “V” and “W” are placed in COs in the metro ring by the HEEDAP algorithm. Hence, the average and maximum RTT experienced by the distributed users of these fog apps is reduced compared to results obtained by solving the MILP model under a similar setup.

6.6 Summary

In this chapter, we extended the application of the composable DC infrastructure paradigm to the edge of the network to improve energy efficiency of the fog computing layer. We considered a fog computing layer which comprised of federated fog computing sites and studied the impact of replacing TSs with DSs at each fog computing site. Our investigation was conducted by formulating a MILP model that energy efficiently placed both interactive and non-interactive fog apps in selected locations. Relative to the use of the TSs in the sites of a fog network that is built over a network with low delay penalty, the adoption of DSs enabled up to 18% reduction in TFPC. Relative to the use TSs in fog network with high delay penalty, the TFPC of the fog network that employed DS architecture was 14% lower. Additionally, our resulted showed that central offices and radio base stations are important edge locations for supporting popular interactive applications. These location are important when energy efficiency is a key design criterion and such applications are moderately sensitive to round-trip delay. Finally, we proposed a heuristic, HEEDAP, which is a real-time algorithm that optimally provisions (delay sensitive) emerging fog applications in a fog network. The HEEDAP algorithm achieved comparable results as those reported when the MILP model was solved under similar scenarios.

Chapter 7 : Conclusions and Future Directions

This chapter gives a summary of the contributions made in this thesis and proposes future directions that may be explored.

7.1 Summary of Contributions

In this work, an investigation is conducted to determine the optimal scale and scope of disaggregation for energy efficient composable DCs under a suitable network. Furthermore, an investigation is also conducted to study the impact of adopting the resource disaggregation in future edge networks and the fog computing tier. Each research problem is formulated at a MILP model and a corresponding heuristic was proposed in some instances.

In Chapter 1, we gave a general introduction on the importance of DCs in supporting both cloud and fog computing paradigms. The emerging trends of adopting resource disaggregation in DC environments to enhance efficiencies was highlighted. We also outlined the objectives and contributions of the work done in this thesis.

Chapter 2 presented a general overview of various components and concepts in this thesis. A comprehensive review of DC infrastructure evolution was made along with a brief review of the software-centric and hardware-centric techniques that were deployed to improve efficiency in DCs. An overview of WDW networks was presented before the virtual topology design problem in WDM networks was briefly reviewed. Lastly, Chapter 2 also reviewed the MILP optimisation approach that is widely used in this thesis.

Chapter 3 of this thesis focused solely on a review of composable DC infrastructure. Firstly, it presented an overview of composable DC infrastructure and subsequently reviewed enabling technologies (i.e., resource disaggregation, software defined infrastructures and optical communication technologies) of composable DC infrastructures. The Chapter also highlighted some benefits and implementation challenges of composable DC infrastructure. Finally, Chapter 3 was concluded by a review of composable DC research in the literature.

In Chapter 4, we formulated a MILP model to evaluate the performance of physical disaggregation of compute resources at rack-scale and pod-scale under selected electrical, optical hybrid network topologies. Relative to resource disaggregation at pod-scale in composable DCs, our results showed that physical disaggregation of compute resources at rack-scale is sufficient

to achieve optimal resource utilisation. This is achieved when appropriate distribution of resource (both in number and/or diversity) is ensured during resource allocation. Adoption of optical network topology in composable DCs ensures optimal overall DC energy efficiency. Physical disaggregation of traditional DC servers at rack-scale leads to better (6%-20%) savings in overall power consumption when memory intensive workloads are provisioned. 5% - 8% savings in total DC power consumption obtained when CPU intensive workloads are provisioned after physical disaggregation of traditional DC servers at rack-scale. Relative to the implementation of physical disaggregation at rack-scale, implementation of logical disaggregation of traditional servers over an optical network topology within a rack ensures a composable infrastructure suitable for all workload categories (i.e., workloads with low-sensitivity or high-sensitivity to increase in memory access latency). Logical disaggregation of traditional servers at rack-scale also improved network power efficiency.

Secondly, Chapter 4 also explored the impact of micro-service architecture on overall DC operational and energy efficiency in both traditional and composable DCs. The results show that disproportionate utilisation of DC resources may persist when micro-service architecture is adopted in a traditional DC. This occurs in spite of improvements in total resource power consumption and resource utilisation enabled by increased workload modularity. Although, disaggregation addresses disproportionate utilisation in traditional DCs, utilisation in disaggregated DCs could be limited when resource intensive monolithic workloads are deployed. Ultimately, a combination of composable infrastructure and increased workload modularity enables optimal resource utilisation and energy efficiencies in DCs. Hence, both approaches are complementary and DC operators should leverage on the strengths of both approaches to enable optimal DC resource utilisation and energy efficiency.

Lastly, we proposed a real-time heuristic for energy efficient placement (HEEP) of workloads in composable DCs in Chapter 4. The total DC power consumption obtained using the HEEP algorithm approached the exact results obtained by solving the MILP model under CPU intensive and memory intensive workload classes. In contrast to the adoption of un-capacitated networks in Chapter 4, network capacity constraints are expected to have significant impact on the practicality and performance of all variants of composable DCs. Hence, the approach adopted in Chapter 4 is somewhat limited in a practical setup.

In Chapter 5, we described two variants of a novel network for composable DCs i.e., NetCoD. We evaluated the performance of the proposed network topologies via a MILP model that performed throughput tests over the novel topologies. Under similar network loads, we found that the electrical-optical variant achieved comparable network energy efficiency as all-optical programmable disaggregated data centre network (AOPD-DCN), a reference topology from the literature. The sole adoption of electrical switches in E-NetCoD led to higher network power consumption. A MILP model was also formulated to evaluate the performance of capacitated networks (i.e., E-NetCoD, EO-NetCoD and AODP-DCN) when energy efficient placement of VMs is required in rack-scale composable DCs. The rack-scale composable DCs considered implement logical, physical and hybrid scopes of resource disaggregation. This addresses the limitation of adopting un-capacitated networks in Chapter 4. The results showed that a practical implementation of the resource disaggregation concept may require marginal concessions in compute energy efficiency to satisfy network constraint. Such concessions are also required to reduce consequential network power consumption that may be incurred when disaggregation in implemented.

Chapter 5 also successfully demonstrated the efficacy of both variants of NetCoD by considering different scopes (i.e., logical, hybrid and physical) of disaggregation in a rack-scale composable DC. The range of scenarios considered highlighted various strategies that can be deployed in practical implementations of either variant of NetCoD to achieve total energy efficiency. Relative to AOPD-DCN, both variants of NetCoD achieved comparable performance in-terms of compute energy efficiency. In-terms of network energy efficiency, EO-NetCoD achieved comparable performance as AOPD-DCN using a targeted and more practical topology design. The targeted design required fewer interfaces at each compute node. E-NetCoD achieved similar performance as EO-NetCoD at higher TNPC because of the high operational power of electrical switches. Hence, making the adoption of electrical switch less desirable in composable DCs that require high energy efficiency. However, the availability of electrical switches (such as the Gen-Z switch and the Ethernet switch proposed for Intel’s rack-scale architecture) which support disaggregation makes E-NetCoD viable option to be considered for composable DC infrastructures. Furthermore, OEO conversion performed by electrical switches can improve the utilisation of the inter-rack network. Moreover, electrical switches are also ideal for practical implementation DC gateway switches. Hence, when deployed in a strategic location as proposed

for EO-NetCoD, an electrical switch could be optimally utilised in a composable DC while minimally impacting network energy efficiency.

Under all network topologies considered in Chapter 5, a logically disaggregated DC achieved the best results. Logically disaggregated DC achieved comparable compute power consumption and compute resource utilisation efficiency as hybrid and physical disaggregation but consumes marginally lower TNPC under all network topologies considered. This is because logical disaggregation enabled optimal use of the highly energy efficient on-board fabric in DCs to significantly minimise the TNPC while achieving high compute energy efficiency as well. Hence, relative to hybrid and physical disaggregation, higher tiers of the DCN are minimally utilised in logically disaggregated DCs. Across all network topologies evaluated in Chapter 5, the average increase in TNPC is 1% and 3.6% when hybrid and physical disaggregation are respectively implemented instead of logical disaggregation. This observation further supports the results obtained in Chapter 4. Furthermore, the results also showed that physical disaggregation significantly exacerbates network capacity constraint, especially for homogenous compute nodes with multiple CPU components. Additionally, utilisation of available network throughput by both variants of NetCoD exceeds that of AOPD-DCN by 4 – 5 times under the different forms of resource disaggregation considered in the rack-scale composable DC. Greater utilisation of available network throughput by both variants of NetCoD is achieved with minimal performance degradation.

Although, theoretically, the resource disaggregation concept has enormous potentials to improve compute energy efficiency in composable DCs, joint recommendations from the investigation conducted in Chapters 4 and 5 include the following:

1. A practical implementation of the resource disaggregation concept, which considers network constraints and power consumption, requires a balance between compute and network energy efficiency. This is needed to achieve an optimal solution that satisfies both compute and network constraints in the DC.
2. Logical disaggregation and rack-scale disaggregation are the recommended scope and scale respectively for energy efficient composable DCs. A combination of both can achieve near optimal compute power consumption and compute resource utilisation efficiency. However, appropriate resource allocation and network in composable infrastructures is required. Furthermore, the combination of logical disaggregation and rack-scale disaggregation significantly minimises network power consumption while maintaining the flexibility expected in composable DCs. Logical disaggregation enables a DC

infrastructure that can support all workloads type. Additionally, increased workload modularity can enhance energy efficiency in composable DCs by complementing resource disaggregation enabled resource modularity.

3. In addition to the enabling low latency and high bandwidth communication, when rightly adopted, optical networking technologies and techniques are important in energy efficient composable DCs. Strategic deployment of optical technologies, with relatively lower power consumption, in composable DCs can minimise the concession required in compute energy efficiency. Furthermore, strategic use of optical technologies and techniques can minimise the complexity and cost of compute node interfaces in composable infrastructures where full-mesh physical connectivity is desirable between co-rack component. This can be achieved by adopting a targeted design approach for composable DC networks as proposed for NetCoD in this thesis.

Chapter 6 applied the recommendations (in Chapters 4 and 5) for energy efficient composable DCs at the edge of the network and in the fog computing tier. We evaluated the energy efficient placement of delay sensitive emerging fog applications in the presence of mission critical traditional fog applications in a shared distributed fog network. The fog network employed traditional server (TS) and disaggregated server (DS) architectures across selected fog computing sites at the network edge. Relative to the use of the TS architecture in the fog network that is built over a network with low delay penalty, the adoption of DSs enabled up to 18% reduction in total fog computing power consumption (TFPC). This is because disaggregation enabled proportional usage of compute resources at each fog computing site. This consequently improved the energy efficiency of the fog network. However, this is achieved at the expense of marginal increase in total network power consumption (TNPC) and somewhat higher response time when DS architecture is adopted. Setting up a fog network with high delay penalty increased the TFPC when either TS or DS architecture are employed in fog computing sites. This was done to minimise the congestion experienced on the network by reducing the network traffic. Consequently, the TNPC is also reduced. But, the TFPC of the fog network that employed DS architecture was 14% lower than that of the fog network that adopted TS architecture. Our result also showed that central offices and radio cell sites are important edge locations for supporting popular interactive applications. This is especially the case when energy efficiency is an important design criterion and such applications are moderately sensitive to the round-trip delay experienced on the network. Otherwise, instances of such fog apps, which are more sensitive to delay,

must be provisioned in the nearest network node that satisfies a predefined (and acceptable) delay threshold to distributed users. Occasional increase in the round-trip delay is experienced by geo-distributed end-users when DSs are deployed in fog network. This may be mitigated by increasing the capacity of the metro and access links at the network edge.

We also proposed a heuristic for energy efficient and delay aware placement (HEEDAP) of applications for a network of distributed fog computing nodes. HEEDAP leverages centralised orchestration and management framework of distributed fog computing nodes. The policy is a real-time algorithm that optimally provisions both mission critical traditional applications and (delay sensitive) emerging fog applications in a fog network when possible. The HEEDAP algorithm achieves comparable results as those reported when the MILP model was solved under similar evaluation scenarios. Often, the HEEDAP algorithm achieved the same application placement pattern, compute and network energy efficiencies as the exact results obtained by solving the MILP model. Occasional difference between the result obtained via by the HEEDAP algorithm and by solving the MILP model are marginal. For example, the difference between the TFPC obtained with the HEEDAP and that obtained by solving the MILP model is not greater than 2% in all scenarios considered.

7.2 Future Directions

The future directions that can be explored as extensions of the works in this thesis include the following.

7.2.1 Resource Component Aggregation

In this thesis, the aggregation of multiple homogenous resource components over the network to satisfy a unique resource demand of a workload was not explored. Future work can study the implementation of such concept in composable DC infrastructure while evaluating the impact of such design on workloads performance. The mathematical models presented in Chapters 4 and 5 of this thesis provide a good foundation that can be extended for such a study.

7.2.2 Adoption of Machine Learning and Artificial Intelligence

Some problems studied in this thesis can be further validated and enhanced using the machine learning and artificial intelligence. In other cases, certain trends identified, from our analysis of MILP model results, in this thesis are suitable for the application of machine learning and artificial intelligence

techniques in a practical setup. Therefore, both directions should be explored to further validate the results in thesis and to enhance practical implementation of results and trends in this thesis. For example, it was observed that intelligence is required to optimally deploy both variants of NetCoD in the DC environment. The reinforcement learning branch of machine learning is a suitable approach that can aid the design of an intelligent agent. Such an intelligent agent would take actions to maximise network utilisation and energy efficiency without compromising performance during workload placement. Reinforcement learning can also be adopted to energy efficiently place fog applications in distributed fog networks. Alternatively, placement of fog applications in the fog network can also be determined using knowledge of user demand distribution and metro and access networks telemetry data. Such information can be fed as input to machine learning models that make energy efficient application placement predictions and inferences.

7.2.3 Multi-Rack Scenario and Empirical Validation for NetCoD

The work in this thesis provided mathematical models for NetCoD and for the energy efficient placement of workloads in a composable DC that deploys NetCoD. However, only the single rack scenario was studied when energy efficient placement of workloads was considered. Conducting a similar study in multi-rack composable DCs could not be achieved due to limited availability of computing power to solve the MILP model for realistic evaluation scenarios. Hence, provided sufficient computational power for a suitable duration, a study that evaluates the performance of NetCoD in multi-rack and multi-cluster scenarios can be conducted. Additionally, an empirical setup can also be created to further validate and verify the performance of NetCoD. Such a setup is more suitable for studies that considers of latency, rate control and traffic differentiation over the converged network relative to the complex mathematical model developed in this thesis. The empirical setup can also provide a suitable platform for the implementation of machine learning and artificial intelligence that can support NetCoD and the placement of workload in DCs that employ NetCoD in realistic scenarios.

7.2.4 Dynamic Fog Applications

In contrast to the static association of certain fog applications with specific fog site as considered in this thesis, future work can consider scenarios where all fog applications can be placed dynamically. Furthermore, inter-workload communication may also be introduced between such applications. Additionally, the HEEDAP algorithm developed in Chapter 6 can be extended

to investigate various placement strategies for dynamic fog applications in metro and access fog computing tiers where composable computing infrastructure is predominantly deployed. In addition, further studies of the impact of user distribution in fog networks can be conducted by introducing a capacity constraint to limit the number of users associated with each provisioned instance of a fog application.

7.2.5 Framework for Energy Efficient Placement in Cloud-Fog Architecture

In this thesis, we have successfully developed a scalable algorithm for energy efficient placement of workloads in composable DCs. We also developed another algorithm for delay aware placement of workloads in metro and access fog networks that employ disaggregated servers. Both algorithms can be combined to develop a holistic framework and policy for the energy efficient deployment of workloads in cloud-fog architecture that spans core, metro, and access networks. Such a framework can provide a tool for a comprehensive study of the impact of joint deployment of composable computing infrastructure in both fog and cloud computing tiers of the cloud-fog architecture. Conducting a similar study of such massive architecture using the MILP model optimisation approach is expected to be challenging because it may not scale well due to high complexity. Combining both algorithms may provide acceptable approximated solutions in such a massive scenario. It is important that such novel framework should also consider the frequency of algorithm re-optimisation. This is because present implementations of the formulated MILP models and proposed algorithms are memoryless. Hence, temporal variation in the distribution of workload demand in DCs is not considered in this thesis. Furthermore, the problems posed by a cloud-fog architecture can also be solved using generic meta-heuristic optimisation methods which are known to be effective and simple. Results obtained via meta-heuristic approaches can be used to validate and improve the effectiveness of the targeted heuristics proposed in this thesis.

7.2.6 Multi-Objective Optimisation

The problem formulations considered in this thesis can be further extended by re-formulating them as multi-objective optimisation problems. For example consideration of the cost and power consumption of DC peripheral components such as cooling and power systems could be added to the objective function. Hence, enabling more realistic representation of practical DCs.

Appendix: List of MILP Notations

Chapter 4 MILP Notations

Sets:

N	Set of nodes of resources
R	Set of racks in the DC
P	Set of Pods in the DC
C	Set of CPU resources
M	Set of memory resources
W	Set of workloads
D	Set of traffic direction
I	Set of integrated workloads

DC Compute Parameters:

$\mathbb{C}_j$	Capacity of CPU module $j \in C$
CP_j	Maximum power consumption of CPU module $j \in C$
IC	Idle power as a fraction of maximum CPU power
ΔC_j	Power factor of CPU module $j \in C$; $\Delta C_j = \frac{CP_j - IC \cdot CP_j}{\mathbb{C}_j}$
$\mathbb{M}_j$	Capacity of memory module $j \in M$
MP_j	Maximum power consumption of memory module $j \in M$
IM	Idle power as a fraction of maximum memory power
ΔM_j	Power factor of memory module $j \in M$; $\Delta M_j = \frac{MP_j - IM \cdot MP_j}{\mathbb{M}_j}$
CN_{jn}	$CN_{jn} = 1$, If CPU $j \in C$ is placed in node $n \in N$. Otherwise $CN_{jn} = 0$
MN_{jn}	$MN_{jn} = 1$ if RAM $j \in M$ is placed in node $n \in N$, Otherwise $MN_{jn} = 0$
NR_{nr}	$NR_{nr} = 1$, If node $n \in N$ is placed in rack $r \in R$, otherwise $NR_{nr} = 0$
RP_{rp}	$RP_{rp} = 1$, If rack $r \in R$ is placed in pod $p \in P$, otherwise $RP_{rp} = 0$
WC_w	CPU capacity required by workload $w \in W$

WM_w Memory capacity required by workload $w \in W$

CI_i CPU capacity of integrated workload i

MI_i Memory capacity of integrated workload i

WI_{wi} Indicates the relationship between a micro-service workload w and integrated workload i. $WI_{wi} = 1$ if micro-service workload w is associated with integrated workload i. Otherwise, $WI_{wi} = 0$

$\mathbb{L}_w$ Maximum latency supported by DC infrastructure for workload $w \in W$

$\mathcal{T}_{cm}$ CPU-Memory latency between CPU component $c \in C$ and memory component $m \in M$. Inter-component latency.

$\mathcal{Q}$ A big number (100000)

$\mathcal{G}$ A big number (1000)

α A weighing factor in Watt which specifies the cost per blocked workload

DC Network Parameters:

$\mathcal{X}$ Static power consumption of optical cross-connect (W)

$\mathcal{W}$ Static power consumption of WSS-based TOR switch (W)

$\mathbb{E}$ Load proportionate energy of electrical switch (J/b)

$\mathcal{E}$ Idle power consumption of electrical switch (W)

$\mathbb{a}$ Number of aggregation switches, $\mathbb{a} \geq 1$; $\frac{\mathbb{a}}{\mathbb{r}}$ is a fixed aggregation ratio.

$\mathbb{b}$ Number of inter-pod cross connects, $\mathbb{b} \geq 1$; $\frac{\mathbb{b}}{\mathbb{p}}$ is a fixed ratio.

$\mathcal{C}_{wx}$ CPU-Memory (RAM) traffic (in b/s) of workload $w \in W$ in direction $x \in D$.

$\mathcal{I}_{wx}$ CPU-IO traffic of workload $w \in W$ in direction $x \in D$.

$\mathcal{R}_{wx}$ Memory(storage)-IO traffic of workload $w \in W$ in direction $x \in D$.

$\mathcal{M}_{sd}$ Inter-memory(storage) traffic between source workload $s \in W$ and destination workload $d \in W$.

EU_{sd} Electrical network load proportional energy per bit (J/b) due to traffic from CPU component $s \in C$ to memory (RAM) component $d \in M$.

ED_{sd} Electrical network load proportional energy per bit (J/b) due to traffic from memory (RAM) component $s \in M$ to CPU component $d \in C$.

EC_{sd} Electrical network load proportional energy per bit (J/b) due to traffic between memory (storage) component $s \in M$ and memory (storage) component $d \in M$.

EN Electrical network load proportional energy per bit (J/b) due to north-south traffic.

HU_{sd} Hybrid network load proportional energy per bit (J/b) due to traffic from CPU component $s \in CR$ to memory (RAM) component $d \in M$.

HD_{sd} Hybrid network load proportional energy per bit (J/b) due to traffic from memory (RAM) component $s \in M$ to CPU component $d \in C$.

HC_{sd} Hybrid network load proportional energy per bit (J/b) due to traffic between memory (storage) component $s \in M$ and memory (storage) component $d \in M$.

HN Hybrid network load proportional energy per bit (J/b) due to north-south traffic.

OU_{sd} Optical network load proportional energy per bit (J/b) due to traffic from CPU component $s \in C$ to memory (RAM) component $d \in M$.

OD_{sd} Optical network load proportional energy per bit (J/b) due to traffic from memory (RAM) component $s \in M$ to CPU component $d \in C$.

OC_{sd} Optical network load proportional energy per bit (J/b) due to traffic between memory (storage) component $s \in M$ and memory (storage) component $d \in M$.

ON Optical network load proportional energy per bit (J/b) due to north-south traffic.

Variables:

$\boldsymbol{c_{wj}}$	$\boldsymbol{c_{wj}} = 1$ indicates that processing requirements of workload $w \in W$ are served by CPU $j \in C$. Otherwise, $\boldsymbol{c_{wj}} = 0$
$\boldsymbol{m_{wj}}$	$\boldsymbol{m_{wj}} = 1$, indicates that memory (RAM) request of workload $w \in W$ is served by RAM $j \in M$. Otherwise, $\boldsymbol{m_{wj}} = 0$
$\mathbb{c}_j$	$\mathbb{c}_j = 1$, if CPU $j \in C$ is active. Otherwise, $\mathbb{c}_j = 0$
$\mathbb{m}_j$	$\mathbb{m}_j = 1$, if RAM $j \in M$ is active. Otherwise, $\mathbb{m}_j = 0$
$\boldsymbol{p_p}$	$\boldsymbol{p_p} = 1$, if pod $p \in P$ is active. Otherwise, $\boldsymbol{p_p} = 0$
$\boldsymbol{r_r}$	$\boldsymbol{r_r} = 1$, if rack $r \in R$ is active. Otherwise, $\boldsymbol{r_r} = 0$
$\mathbb{w}_w$	Indicates the state of workload $w \in W$ i.e., served, or unserved.$\mathbb{w}_w = 1$, if workload w is served. Otherwise, $\mathbb{w}_w = 0$
$\boldsymbol{\beta_w}$	Indicates the state of workload $w \in W$ i.e., rejected, or active. It is the opposite of $\mathbb{w}_w$, $\boldsymbol{\beta_w} = 1 - \mathbb{w}_w$
$\mathbb{i}_i$	Indicates the state of integrated workload $i \in I$ i.e., served, or unserved. $\mathbb{i}_i = 1$ indicates the integrated workload i is served. Otherwise, $\mathbb{i}_i = 0$
$\mathbb{r}$	Number of active racks
$\mathbb{p}$	Number of active pods
$\boldsymbol{h_{wr}}$	$\boldsymbol{h_{wr}} = 1$, if CPU resource demand of workload $w \in W$ is placed in rack $r \in R$. Otherwise, $\boldsymbol{h_{wr}} = 0$
$\boldsymbol{g_{wr}}$	$\boldsymbol{g_{wr}} = 1$, if memory resource demand of workload $w \in W$ is placed in rack $r \in R$. Otherwise, $\boldsymbol{g_{wr}} = 0$
$\boldsymbol{a_{wp}}$	$\boldsymbol{a_{wp}} = 1$, if CPU resource demand of workload $w \in W$ is placed in pod $p \in P$. Otherwise, $\boldsymbol{a_{wp}} = 0$
$\boldsymbol{b_{wp}}$	$\boldsymbol{b_{wp}} = 1$, if memory resource demand of workload $w \in W$ is placed in pod $p \in P$. Otherwise, $\boldsymbol{b_{wp}} = 0$
$\boldsymbol{y_{wcm}}$	Indicates the CPU-memory pair used to provision workload $w \in W$.$\boldsymbol{y_{wcm}}$= 1 if CPU $c \in C$ and memory $m \in M$ host CPU and memory resource demands of workload $w \in W$ respectively. Otherwise, $\boldsymbol{y_{wcm}}$ = 0.
$\boldsymbol{z_{sd}^{xy}}$	$\boldsymbol{z_{sd}^{xy}} = 1$ if memory resource demand of source workload $s \in W$ is placed in memory component $x \in M$ and memory

resource demand of destination workload $d \in W$ is placed in memory component $y \in M$. Otherwise, $\boldsymbol{z_{sd}^{xy}} = 0$

Chapter 5 MILP Notations

Sets:

A	Set of compute nodes.
G	Set of DC gateway switches to the Internet
Y	Set of compute nodes and DC gateway switches
Z	Set of leaf and spine switches
Q	Set of routing and forwarding nodes in the DC
N	Set of all Nodes
N_m	Set of all neighbour nodes of node $m \in N$; $N_m \subseteq N$.
B_m	Set of all intra-rack neighbour nodes of node $m \in N$; $B_m \subseteq N$.
A_m	Set of all compute nodes that are neighbours of compute node $m \in A$; $A_m \subseteq A$.
O	Set of transmission wavelengths supported in the network.
T	Set of interfaces supported by a compute node.
X	Set of optical switches, $X \subseteq N$
H_m	Set of all neighbour nodes of node $m \in N$; $H_m \subseteq N$ which are part of the hybrid inter-rack network.
C	Set of CPU resource components
M	Set of memory resource components
S	Set of storage resource components
R	Set of DC racks

Parameters:

$\mathbb{T}_{of}$	$\mathbb{T}_{of} = 1$ if wavelength $o \in O$ is allocated to interface $f \in T$ for transmission of data traffic, otherwise $\mathbb{T}_{of} = 0$
$\mathbb{R}_{of}$	$\mathbb{R}_{of} = 1$ if wavelength $o \in O$ is allocated to interface $f \in T$ for reception of data traffic, otherwise $\mathbb{R}_{of} = 0$

μ_i	Load proportional routing cost for a routing and forwarding node $i \in Q$, (J/b)
OB	On-board network interface energy per bit (J/b)
TX_m	Transmitting energy per bit (J/b) of routing and forwarding node $m \in Q$
RX_m	Receiving energy per bit (J/b) of routing and forwarding node $m \in Q$
RL_m	Relaying energy per bit (J/b) of routing and forwarding node $m \in Q$
$\mathcal{X}$	Optical switch operational power in Watt
$\mathcal{E}$	Electrical switch operational power in Watt
$\mathcal{S}$	SOA switch energy per bit (J/b)
ρ	Number of spine switches in the composable DC
ϱ	Number of electrical gateway or super-spine switches in the composable DC
$\mathbb{r}$	Number of active racks in the composable DC
τ_{sd}	Total traffic from node $s \in Q$ to node $d \in Q$.
$\mathcal{D}$	Maximum data rate of a single wavelength.
$\mathbb{D}$	Maximum transmitting and receiving capacity supported on the point-to-point physical link between nodes on the intra-rack backplane in AOPD-DCN
$\mathcal{Q}$	A big number (100000)
$\mathcal{G}$	A big number (1000)
κ	Cost associated with each path provisioned in an optical switch in Watt
$\mathbb{C}_j$	Capacity of CPU component $j \in C$
IC	Idle power as a fraction of maximum CPU power consumption
CP_j	Maximum power consumption of CPU component $j \in C$
ΔC_j	Power factor of CPU component $j \in C$ in W/GHz
$\mathbb{M}_j$	Capacity of memory component $j \in M$

IM	Idle power as a fraction of maximum memory power consumption
MP_j	Maximum power consumption of memory component $j \in M$
ΔM_j	Power factor of memory component $j \in M$ in W/GB
$\mathbb{S}_j$	Capacity of storage component $j \in S$
IS	Idle power as a fraction of maximum storage power
SP_j	Maximum power consumption of storage component $j \in S$
ΔS_j	Power factor of storage component $j \in S$ in W/GB
CN_{jn}	$CN_{jn} = 1$ if CPU $j \in C$ is placed in node $n \in N$. Otherwise $CN_{jn} = 0$. Note that CPU components can only be placed in compute nodes.
MN_{jn}	$MN_{jn} = 1$ if RAM $j \in M$ is placed in node $n \in N$. Otherwise $MN_{jn} = 0$. Note that memory components can only be placed in compute nodes
SN_{jn}	$SN_{jn} = 1$ if hard disk drive (HDD) $j \in S$ is placed in node $n \in N$. Otherwise $SN_{jn} = 0$. Note that storage components can only be placed in compute nodes
NR_{nr}	$NR_{nr} = 1$, If node $n \in N$ is placed in rack $r \in R$, otherwise $NR_{nr} = 0$
VC_v	CPU demand of VM $v \in V$
VM_v	RAM demand of VM $v \in V$
VS_v	Storage demand of VM $v \in V$
$\mathcal{CU}_v$	CPU to memory (RAM) traffic required by VM $v \in V$
$\mathcal{CD}_v$	Memory (RAM) to CPU traffic required by VM $v \in V$
$\mathcal{SU}_v$	CPU to storage traffic required by VM $v \in V$
$\mathcal{SD}_v$	Storage to CPU traffic required by VM $v \in V$
$\mathcal{IU}_v$	Uplink north-south traffic of VM $v \in V$
$\mathcal{ID}_v$	Downlink north-south traffic of VM $v \in V$
$\mathcal{M}_{sd}$	In-memory computing traffic from VM $s \in V$ to VM $d \in V$
VG_{vn}	$VG_{vn} = 1$ denotes that node $n \in G$ is the gateway node for north-south traffic of VM $v \in V$

α Cost associated with a VM rejection.

Variables

$\mathbf{T_{sd}}$ Total traffic from node $s \in \mathrm{Q}$ to node $d \in \mathrm{Q}$.

$\mathbf{T_{sd}^{ij}}$ Volume of T_{sd} traversing virtual link (i,j). $i \in \mathrm{Q}, j \in Q, s \in \mathrm{Q}, d \in \mathrm{Q}: i \neq j, s \neq d$. It denotes routing of traffic in the virtual network.

$\boldsymbol{v_{ij}}$ Volume of traffic on virtual link (i,j); $i \in Q, j \in \mathrm{Q}$

$\boldsymbol{\Phi_i}$ Traffic transmitted at routing node $i \in Q$

$\boldsymbol{\Psi_i}$ Traffic received at routing node $i \in Q$

$\boldsymbol{\Omega_i}$ Traffic relayed at routing node $i \in Q$

$\boldsymbol{\phi_m}$ Traffic transmitted at forwarding node $m \in Q$

$\boldsymbol{\psi_m}$ Traffic received at forwarding node $m \in Q$

$\boldsymbol{\omega_m}$ Traffic relayed at forwarding node $m \in Q$

$\boldsymbol{v_{omn}^{ij}}$ Volume of traffic on virtual link (i,j) using wavelength $o \in O$ on physical link $(m,n), i \in Q, j \in Q, m \in N, n \in N_m: i \neq j, m \neq n$

$\boldsymbol{w_{omn}}$ Volume of traffic using wavelength $o \in O$ on physical link (m,n), $m \in N, n \in N_m: m \neq n$

$\boldsymbol{q_{omn}}$ $\boldsymbol{q_{omn}} = 1$ if $\boldsymbol{w_{omn}} > 0$. Otherwise $\boldsymbol{q_{omn}} = 0, o \in O, m \in N, n \in B_m: m \neq n$

$\boldsymbol{g_{omn}}$ $\boldsymbol{g_{omn}} = 1$, if $\boldsymbol{q_{omn}} \vee \boldsymbol{q_{onm}} = 1$. Otherwise $\boldsymbol{g_{omn}} = 0, o \in O, m \in N, n \in B_m: m \neq n$

$\boldsymbol{e_{ofm}}$ $\boldsymbol{e_{ofm}} = 1$ if wavelength $o \in O$ is used on interface $f \in T$ of compute node $m \in \mathrm{A}$ either to transmit traffic to neighbour nodes or receive traffic from neighbour nodes. Otherwise, $\boldsymbol{e_{ofm}} = 0$.

$\boldsymbol{u_{ij}}$ Volume of traffic on virtual link (i,j), $i \in Q, j \in Q$, that traverses intra-rack network.

$\boldsymbol{\ell_{oij}}$ Volume of traffic using wavelength $o \in O$ on virtual link (i,j), $i \in Q, j \in Q$, that traverses the hybrid inter-rack network.

$\boldsymbol{\mathcal{L}_{oij}}$ $\boldsymbol{\mathcal{L}_{oij}} = 1$ if $\boldsymbol{\ell_{oij}} > 0$. Otherwise, $\boldsymbol{\mathcal{L}_{oij}} = 0, o \in O, i \in Q, j \in Q: i \neq j$

$\boldsymbol{x_{omxn}}$	The configured switching matrix of an optical switch. $\boldsymbol{x_{omxn}} = 1$ if the wavelength $o \in O$ from node $m \in H_x$ enters optical switch $x \in X$ and is relayed to node $n \in H_x$. Otherwise, $\boldsymbol{x_{omxn}} = 0$.
$\boldsymbol{d_{omxn}^{ij}}$	$\boldsymbol{d_{omxn}^{ij}}$ gives the traffic $\boldsymbol{v_{omx}^{ij}}$ that enters optical switch $x \in X: x \in H_m$ from node $m \in H_x$ and is relayed to node $n \in H_x$ on the hybrid inter-rack network. $o \in O, m \in N$
$\boldsymbol{c_{vj}}$	$\boldsymbol{c_{vj}} = 1$ indicates that CPU demand of VM $v \in V$ is served by CPU $j \in C$. Otherwise, $\boldsymbol{c_{vj}} = 0$
$\boldsymbol{m_{vj}}$	$\boldsymbol{m_{vj}} = 1$ indicates that RAM demand of VM $v \in V$ is served by memory (RAM) $j \in M$. Otherwise, $\boldsymbol{m_{vj}} = 0$
$\boldsymbol{s_{vj}}$	$\boldsymbol{s_{vj}} = 1$ indicates that storage resource demand of VM $v \in V$ is served by HDD $j \in S$. Otherwise, $\boldsymbol{s_{vj}} = 0$
$\mathbb{c}_j$	$\mathbb{c}_j = 1$ if CPU $j \in C$ is active. Otherwise, $\mathbb{c}_j = 0$
$\mathbb{m}_j$	$\mathbb{m}_j = 1$ if RAM $j \in M$ is active. Otherwise, $\mathbb{m}_j = 0$
$\mathbb{s}_j$	$\mathbb{s}_j = 1$ if HDD $j \in S$ is active. Otherwise, $\mathbb{s}_j = 0$
$\mathbb{a}_n$	$\mathbb{a}_n = 1$ if compute node $n \in A$ is active. Otherwise, $\mathbb{a}_n = 0$
$\boldsymbol{r_r}$	$\boldsymbol{r_r} = 1$ if rack $r \in R$ is active. Otherwise, $\boldsymbol{r_r} = 0$
$\mathbb{r}$	Number of active racks in the composable DC
$\boldsymbol{CM_{vsd}}$	$\boldsymbol{CM_{vsd}} = 1$ if CPU demand of VM $v \in V$ is placed in compute node $s \in A$ and mermory demand of VM $v \in V$ is placed in compute node $d \in A$. Otherwise, $\boldsymbol{CM_{vsd}} = 0$.
$\boldsymbol{CS_{vsd}}$	$\boldsymbol{CS_{vsd}} = 1$ if CPU demand of VM $v \in V$ is placed in compute node $s \in A$ and storage demand of VM $v \in V$ is placed in compute node $d \in A$. Otherwise, $\boldsymbol{CS_{vsd}} = 0$.
$\boldsymbol{z_{sd}^{xy}}$	$\boldsymbol{z_{sd}^{xy}} = 1$ if memory to memory (in-memory computing) traffic exists from VM $x \in V$ in compute node $s \in A$ to VM $y \in V$ in compute node $d \in A$. Otherwise, $\boldsymbol{z_{sd}^{xy}} = 0$.
$\boldsymbol{IR_{sd}}$	Total inter-resource traffic from compute node $s \in A$ to compute node $d \in A$ due to VM resource demand placement.
$\boldsymbol{EW_{sd}}$	Total east-west traffic from node $s \in N$ to node $d \in N$.
$\boldsymbol{NS_{sd}}$	Total north-south traffic from node $s \in N$ to node $d \in N$.
$\mathbf{T}_{sd}$	Total traffic from node $s \in N$ to node $d \in N$.

$\boldsymbol{\beta_v}$	$\boldsymbol{\beta_v} = 1$ if VM $v \in V$ is rejected. Otherwise, $\boldsymbol{\beta_v} = 0$.
$\boldsymbol{\beta}$	Total number of rejected VMs

Chapter 6 MILP Notations

Sets:

N	Set of all network nodes
NB_m	Set of neighbour nodes of network node $m \in N$
U	Set of metro network nodes, $U \subseteq N$
U_m	Set of neighbour metro nodes of metro node $m \in U$
G	Set of gateway nodes in the metro network topology, $G \subseteq U$
AN	Set of access network nodes $AN \subseteq N$
AN_a	Set of metro nodes that are neighbours of access network node $a \in AN$
F	Set of fog apps
TA	Set of traditional fog apps, $TA \subseteq F$
E	Set of emerging fog apps, $E \subseteq F$
C	Set of CPU resource components.
M	Set of memory resource components.
S	Set of storage resource components.
A	Set of computing nodes

Parameters:

AC_a	$AC_a = 1$ if access network node $a \in AN$ has a consumer premises equipment. Otherwise, $AC_a = 0$.
AP_a	$AP_a = 1$ if access network node $a \in AN$ has a PON ONU. Otherwise, $AP_a = 0$.
PL_{mn}	Bandwidth of physical link (m,n) $m \in N, n \in NB_m$
Δ_{an}	$\Delta_{an} = 1$ if node $n \in N$ is an access network node $a \in AN$. Otherwise, $\Delta_{an} = 0$
RT_{mn}	Regular traffic on physical link (m,n) $m \in N, n \in NB_m$
PD_{mn}	Propagation delay on physical link (m,n) $m \in N, n \in NB_m$

LP_{mn}	Set of linear pieces (linear approximations) used to linearise the delay curve of the delay experienced on link (m,n) $m \in N, n \in NB_m$
∇_{mnq}	Rate of linear piece $q \in LP_{mn}$ of the linear approximation of the delay experienced on link (m,n) $m \in N, n \in NB_m$
ζ_{mnq}	Intercept of linear piece $q \in LP_{mn}$ of the linear approximation of the delay experienced on link (m,n) $m \in N, n \in NB_m$
LU_{mn}	Upper bound of queuing delay experienced on link (m,n) $m \in N, n \in NB_m$
CE	Consumer premises equipment power consumption
MA	Metro Ethernet access switch energy per bit
MG	Metro Ethernet aggregation switch energy per bit
NU	PON ONU power consumption
OL	PON OLT energy per bit
δ	Queuing penalty of the network topology in Watt per second.
FC_f	Compute resource demand of fog app $f \in F$
FM_f	Memory resource demand of fog app $f \in F$
FS_f	Storage resource demand of fog app $f \in F$
FU_e	Uplink data rate per user of emerging fog app $e \in E$
FD_e	Downlink data rate per user of emerging fog app $e \in E$
TS_{tn}	$TS_{tn} = 1$ if traditional fog app $t \in TA$ is associated with network node $n \in N$. Otherwise, $TS_{tn} = 0$
EG_{en}	$EG_{en} = 1$ if node $n \in G$ is the gateway node of emerging fog app $e \in E$. Otherwise, $EG_{en} = 0$
EA_{ea}	$EA_{ea} = 1$ if users in node $a \in AN$ make request for an instance of emerging fog app $e \in E$. Otherwise, $EA_{ea} = 0$.
UN_{ea}	Number of users in node $a \in AN$ requesting an instance of emerging fog app $e \in E$.
ED_e	Emerging fog app $e \in E$ maximum delay threshold.
γ	Cost coefficient of power consumed as a result of rejecting traditional fog apps.

$\emptyset$	Cost coefficient of power consumed as a result of rejecting emerging fog apps.
$\mathbb{C}_j$	Capacity of CPU component $j \in C$
IC	Idle power consumption as a fraction of the maximum CPU power consumption.
CP_j	Maximum power consumption of CPU $j \in C$
ΔC_j	Power factor of CPU $j \in C$; $\Delta C_j = \frac{CP_j - IC \cdot CP_j}{\mathbb{C}_j}$
$\mathbb{M}_j$	Capacity of memory component $j \in M$
IM	Idle power consumption as a fraction of the maximum memory power consumption.
MP_j	Maximum power consumption of memory $j \in M$
ΔM_j	Power factor of memory $j \in M$; $\Delta M_m = \frac{MP_j - IM \cdot MP_j}{\mathbb{M}_j}$
$\mathbb{S}_j$	Capacity of storage component $j \in S$
IS	Idle power consumption as a fraction of the maximum storage power consumption.
SP_j	Maximum power consumption of storage $j \in S$
ΔS_j	Power factor of storage $j \in S$; $\Delta S_j = \frac{SP_j - IS \cdot SP_j}{\mathbb{S}_j}$
$M\Delta C$	Power factor of CPU component with highest power consumption.
CPM	Maximum power consumption of CPU component with highest power consumption.
$M\Delta M$	Power factor of memory component with highest power consumption.
MPM	Maximum power consumption of memory component with highest power consumption.
$M\Delta S$	Power factor of storage component with highest power consumption.
SPM	Maximum power consumption of storage component with highest power consumption.
CN_{cx}	$CN_{cx} = 1$ if CPU component $c \in C$ is placed in computing node $x \in A$. Otherwise, $CN_{cx} = 0$.

MN_{mx} $MN_{mx} = 1$ if memory component $m \in M$ is placed in computing node $x \in A$. Otherwise, $MN_{mx} = 0$.

SN_{sx} $SN_{sx} = 1$ if storage component $s \in S$ is placed in compute node $x \in A$. Otherwise, $SN_{sx} = 0$.

AM_{xn} $AM_{xn} = 1$ if compute node $x \in A$ is placed in network node $n \in N$. Otherwise, $AM_{xn} = 0$.

Q A large enough number.

Variables:

$\boldsymbol{c_{fc}}$ $\boldsymbol{c_{fc}} = 1$ if an instance of fog app $f \in F$ is in CPU component $c \in C$. Otherwise, $\boldsymbol{c_{fc}} = 0$.

$\boldsymbol{m_{fm}}$ $\boldsymbol{m_{fm}} = 1$ if an instance of fog app $f \in F$ is in memory component $m \in M$. Otherwise, $\boldsymbol{m_{fm}} = 0$.

$\boldsymbol{s_{fs}}$ $\boldsymbol{s_{fs}} = 1$ if an instance of fog app $f \in F$ is in storage component $s \in S$.Otherwise, $\boldsymbol{s_{fs}} = 0$.

$\mathbb{c}_{\boldsymbol{c}}$ $\mathbb{c}_{\boldsymbol{c}} = 1$ if CPU $c \in C$ is active. Otherwise, $\mathbb{c}_{\boldsymbol{c}} = 0$.

$\mathbb{m}_{\boldsymbol{m}}$ $\mathbb{m}_{\boldsymbol{m}} = 1$ if memory $m \in M$ is active. Otherwise,$\mathbb{m}_{\boldsymbol{m}} = 0$.

$\mathbb{s}_{\boldsymbol{s}}$ $\mathbb{s}_{\boldsymbol{s}} = 1$ if storage $s \in S$ is active. Otherwise, $\mathbb{s}_{\boldsymbol{s}} = 0$.

$\mathbb{x}_{\boldsymbol{eca}}$ $\mathbb{x}_{\boldsymbol{eca}} = 1$ if the instance of emerging fog app $e \in E$ in CPU component $c \in C$ is allocated to users in access node $a \in AN$. Otherwise, $\mathbb{x}_{\boldsymbol{eca}} = 0$.

$\mathbb{v}_{\boldsymbol{ea}}$ $\mathbb{v}_{\boldsymbol{ea}} = 1$ if the emerging app $e \in E$ requested by node $a \in AN$ has been provisioned. Otherwise $\mathbb{v}_{\boldsymbol{ea}} = 0$.

$\boldsymbol{\varphi_{esa}}$ $\boldsymbol{\varphi_{esa}} \geq 1$ if an instance of emerging fog app $e \in E$ in node $s \in N$ is allocated to users of that app in access node $a \in AN$. Otherwise, $\boldsymbol{\varphi_{esa}} = 0$.

$\mathbb{k}_{\boldsymbol{sd}}$ Post-processing traffic from instances of all fog apps in network node $s \in N$ to gateway node $d \in G$.

$\mathbb{y}_{\boldsymbol{sdec}}$ Pre-processing traffic from users in node $s \in N$ to node $d \in N$ that hosts an instance of emerging fog app $e \in E$ placed in CPU component $c \in C$ in node $d \in N$.

$\mathbb{z}_{\boldsymbol{sdec}}$ Post-processing traffic from compute node $s \in N$ to users in node $d \in N$. Node $s \in N$ hosts an instance of emerging fog app $e \in E$ placed in CPU component $c \in C$. The instance

	of emerging fog app $e \in E$ placed in CPU component $c \in C$ was allocated to users in node $d \in N$.
$\boldsymbol{L_{sdec}}$	Traffic from node $s \in N$ to node $d \in N$ due to the presence of emerging fog app $e \in E$ in CPU component $c \in C$.
$\mathbb{h}_{mn}^{sd}$	Volume of $\mathbb{k}_{sd}$ traffic routed on physical link (m, n)
$\boldsymbol{\lambda_{mn}}$	Volume of cloud bound traffic on physical link (m, n).
$\boldsymbol{\mathcal{H}_{mn}^{sdec}}$	Flow of latency sensitive traffic (emerging fog applications traffic) $\boldsymbol{L_{sdec}}$ on physical link (m, n)
$\mathbb{H}_{mn}^{sdec}$	$\mathbb{H}_{mn}^{sdec} = 1$ if a flow of $\boldsymbol{L_{sdec}}$ is present on physical link (m, n). Otherwise $\mathbb{H}_{mn}^{sdec} = 0$.
$\boldsymbol{\Lambda_{mn}}$	Volume of latency sensitive traffic on physical link (m, n)
$\boldsymbol{\Gamma_{mn}}$	Total traffic on physical link (m, n)
$\mathbb{u}_{m}$	The traffic relayed by a metro node $m \in U$
$\mathbb{g}_{m}$	The traffic received by a metro node $m \in U$
$\mathbb{q}_{m}$	The fog transmitted by a metro node $m \in U$
$\mathbb{t}_{t}$	State of traditional fog app $t \in TA$.
$\boldsymbol{TCRTA}$	Total cost of rejected traditional fog apps in Watt.
$\boldsymbol{TCREA}$	Total cost of rejected emerging fog app in Watt.
$\boldsymbol{\alpha_t}$	Power penalty as a result of rejecting traditional fog app $t \in TA$ in Watt.
$\boldsymbol{\beta_e}$	Power penalty as a result of rejecting emerging fog app $e \in E$ in Watt.
$\mathbb{W}_{mn}$	M/M/1 queuing delay experienced on physical link (m, n)
$\boldsymbol{TL_{mn}^{sdec}}$	$\boldsymbol{TL_{mn}^{sdec}}$ is the queuing delay experienced by flow $\boldsymbol{L_{sdec}}$ on physical link (m, n) on the path selected for the flow.
$\boldsymbol{PD_{mn}^{sdec}}$	$\boldsymbol{PD_{mn}^{sdec}}$ is the propagation delay experienced by flow $\boldsymbol{L_{sdec}}$ on physical link (m, n) on the path selected for the flow.
$\boldsymbol{WL_{sdec}}$	Total delay of flow $\boldsymbol{L_{sdec}}$ on all physical links. Sum of congestion in physical links (queuing delay) and propagation delay on physical links on the path.

$\boldsymbol{RD_{sdec}}$	Round trip delay between a node containing users of an emerging fog app and the network node hosting the instance assigned to the users.
$\boldsymbol{TQ}$	Approximated total queuing delay experienced on physical links of the network topology.

List of References

[1] LogicMonitor Inc, “Cloud Vision 2020: The Future of Cloud,” 2017. [Online]. Available: https://www.logicmonitor.com/wp-content/uploads/2017/12/LogicMonitor-Cloud-2020-The-Future-of-the-Cloud.pdf. [Accessed: 17-Jun-2019].

[2] IDC, “IDC Forecasts Worldwide Technology Spending on the Internet of Things to Reach $1.2 Trillion in 2022,” 2018. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS43994118. [Accessed: 17-Jun-2019].

[3] Cisco, “Internet of Things,” 2016. [Online]. Available: www.cisco.com/go/iot. [Accessed: 17-Jun-2019].

[4] Cisco, “Cisco Global Cloud Index: Forecast and Methodology, 2015–2020,” *Cisco Global Cloud Index*, 2016. [Online]. Available: http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf. [Accessed: 15-May-2017].

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The Case for VM-Based Cloudlets in Mobile Computing,” *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog Computing and Its Role in the Internet of Things,” in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16.

[7] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “‘Fog Computing: A Platform for Internet of Things and Analytics’ Flavio Bonomi, Rodolfo Milito, Preethi Natarajan and Jiang Zhu, N. Bessis and C. Dobre (eds.), Big Data and Internet of Things: 169 A Roadmap for Smart Environments, Studies in Computational,” 2014.

[8] M. Aazam, S. Zeadally, and K. A. Harras, “Fog Computing Architecture, Evaluation, and Future Research Directions,” *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 46–52, 2018.

[9] Y. Yang, “FA2ST: Fog as a Service Technology,” in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 2017, vol. 1, p. 708.

[10] W. Li *et al.*, “System modelling and performance evaluation of a three-tier Cloud of Things,” *Futur. Gener. Comput. Syst.*, vol. 70, pp. 104–125, 2017.

[11] Cisco, “Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are What You Will Learn,” 2015.

[12] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, “On Reducing IoT Service Delay via Fog Offloading,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, 2018.

[13] J. Taylor, “Facebook’s data center infrastructure: Open compute, disaggregated rack, and beyond,” *2015 Optical Fiber Communications Conference and Exhibition (OFC)*. p. 1, 2015.

[14] A. D. Papaioannou, R. Nejabati, and D. Simeonidou, “The Benefits of a Disaggregated Data Centre: A Resource Allocation Approach,” *2016 IEEE Global Communications Conference (GLOBECOM)*. pp. 1–7, 2016.

[15] B. Abali, R. J. Eickemeyer, H. Franke, C.-S. Li, and M. Taubenblatt, “Disaggregated and optically interconnected memory: when will it be cost effective?,” *CoRR*, vol. abs/1503.0, 2015.

[16] H. M. Mohammad Ali, T. E. H. El-Gorashi, A. Q. Lawey, and J. M. H. Elmirghani, “Future Energy Efficient Data Centers With Disaggregated Servers,” *J. Light. Technol.*, vol. 35, no. 24, pp. 5361–5380, 2017.

[17] R. Lin, Y. Cheng, M. De Andrade, L. Wosinska, and J. Chen, “Disaggregated Data Centers: Challenges and Trade-offs,” *IEEE Commun. Mag.*, vol. 58, no. 2, pp. 20–26, Feb. 2020.

[18] C. Kachris and I. Tomkos, “A Survey on Optical Interconnects for Data Centers,” vol. 14, no. 4, pp. 1021–1036, 2012.

[19] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, “DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers,” in *SIGCOMM08*, 2008.

[20] T. Benson, A. Akella, and D. A. Maltz, “Network Traffic Characteristics of Data Centers in the Wild,” in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 267–280.

[21] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The Cost of a Cloud: Research Problems in Data Center Networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[22] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *Computer Communication Review*, 2008, vol. 38, no. 4, pp. 63–74.

[23] N. Farrington and A. Andreyev, “Facebook’s data center network architecture,” *2013 Opt. Interconnects Conf.*, pp. 49–50, 2013.

[24] Cisco, “Cisco Data Center Spine-and-Leaf Architecture: Design Overview White Paper - Cisco,” 2016. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html. [Accessed: 14-Feb-2017].

[25] R. Sohan, A. Rice, W. M. Andrew, and K. Mansley, “Characterizing 10 Gbps network interface energy consumption,” in *IEEE Local Computer Network Conference*, 2010, pp. 268–271.

[26] C. Guo *et al.*, “BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers,” in *ACM SIGCOMM*, 2009, pp. 63–74.

[27] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, “FiConn : Using Backup Port for Server Interconnection in Data Centers,” pp. 2276–2285, 2009.

[28] G. Wang *et al.*, “C-Through: Part-Time Optics in Data Centers,” *Proc. ACM SIGCOMM 2010 Conf.*, vol. 40, no. 4, pp. 327–338, Aug. 2010.

[29] N. Farrington *et al.*, “Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers,” in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 339–350.

[30] K. Chen *et al.*, “OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 498–511, Apr. 2014.

[31] G. M. Saridis *et al.*, “Lightness: A Function-Virtualizable Software Defined Data Center Network With All-Optical Circuit/Packet Switching,” *Journal of Lightwave Technology*, vol. 34, no. 7. pp. 1618–1627, 2016.

[32] X. Ye, P. Mejia, Y. Yin, R. Proietti, S. J. B. Yoo, and V. Akella, “DOS - A scalable optical switch for datacenters,” *2010 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. pp. 1–12, 2010.

[33] A. A. Hammadi, “Future PON Data Centre Networks,” University of Leeds, Leeds, United Kingdom, 2016.

[34] A. Roozbeh *et al.*, “Software-Defined ‘Hardware’ Infrastructures: A Survey on Enabling Technologies and Open Research Directions,” *IEEE Commun. Surv. Tutorials*, pp. 1–1, 2018.

[35] IBM Cloud Education, “Containerization,” 15-May-2019. [Online]. Available: https://www.ibm.com/cloud/learn/containerization. [Accessed: 19-Jul-2020].

[36] Kubernetes, “What is Kubernetes?” [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. [Accessed: 19-Jul-2020].

[37] Microsoft, “Introduction to microservices on Azure | Microsoft Docs.” [Online]. Available: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview-microservices. [Accessed: 22-Apr-2018].

[38] J. Lewis and M. Fowler, “Microservices,” 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html. [Accessed: 03-Sep-2017].

[39] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to Cloud-Native Architectures Using Microservices: An Experience Report,” in *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers*, A. Celesti and P. Leitner, Eds. Cham: Springer International Publishing, 2016, pp. 201–215.

[40] R. V. Shahir Daya Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo martins, Shishir Narain, *Microservices from Theory to Practice Creating Applications in IBM Bluemix Using the Microservices Approach*, First. IBM Redbooks, 2015.

[41] A. Sill, “The Design and Architecture of Microservices,” *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 76–80, 2016.

[42] S. Miniman, “The Data Center: Past, Present And Future,” 06-Feb-

2014. [Online]. Available: http://wikibon.org/wiki/v/The_Data_Center:_Past,_Present_and_Future. [Accessed: 17-Jul-2020].

[43] Amazon Web Services, “Serverless Computing.” [Online]. Available: https://aws.amazon.com/serverless/. [Accessed: 19-Jul-2020].

[44] Microsoft Azure, “Serverless computing .” [Online]. Available: https://azure.microsoft.com/en-gb/overview/serverless-computing/. [Accessed: 19-Jul-2020].

[45] Google Cloud, “Serverless computing.” [Online]. Available: https://cloud.google.com/serverless. [Accessed: 19-Jul-2020].

[46] Google Cloud, “What Is Serverless Architecture, and How Does It Help My Business?” [Online]. Available: https://lp.google-mkto.com/rs/248-TPC-286/images/GC-Article-What-Is-Serverless.pdf?utm_campaign=2016-gc-cnc-nurture-email-other-a3-vpinfra-infra&utm_source=nurture&utm_medium=emailinfra3-3b&gcpngmt=infra&cntctg=awrns&mkt_tok=eyJpIjoiWkRsaU0yUTNaamRtTXpkbS. [Accessed: 05-Sep-2018].

[47] M. Roberts, “Serverless Architectures,” 22-May-2018. [Online]. Available: https://martinfowler.com/articles/serverless.html. [Accessed: 19-Jul-2020].

[48] A. Andrae and T. Edler, “On Global Electricity Usage of Communication Technology: Trends to 2030,” *Challenges*, vol. 6, no. 1, pp. 117–157, Apr. 2015.

[49] Facebook, “Designing a Very Efficient Data Center,” 14-Apr-2011. [Online]. Available: https://www.facebook.com/notes/facebook-engineering/designing-a-very-efficient-data-center/10150148003778920/. [Accessed: 20-Jul-2020].

[50] Google, “Machine learning finds new ways for our data centers to save energy.” [Online]. Available: https://sustainability.google/projects/machine-learning/. [Accessed: 20-Jul-2020].

[51] Open Compute Project, “About.” [Online]. Available: https://www.opencompute.org/about. [Accessed: 20-Jul-2020].

[52] Facebook, “Building Efficient Data Centers with the Open Compute Project,” 07-Apr-2011. [Online]. Available: https://www.facebook.com/note.php?note_id=10150144039563920. [Accessed: 20-Jul-2020].

[53] Intel Corporation, “Intel® Rack Scale Design Is Now Ready for Open Source Development - IT Peer Network,” 2016. [Online]. Available: https://itpeernetwork.intel.com/intel-rack-scale-design-now-ready-open-source-development/. [Accessed: 15-May-2017].

[54] Intel, “Intel® RSD Partners.” [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design/rack-scale-design-partners.html. [Accessed: 20-Jul-2020].

[55] European Telecommunications Standards Institute, “Network Functions Virtualization,” 2018. [Online]. Available: www.etsi.org. [Accessed: 29-Mar-2019].

[56] Facebook, “On Our Way to Lower Emissions and 100% Renewable Energy,” 28-Aug-2018. [Online]. Available: https://about.fb.com/news/2018/08/renewable-energy/. [Accessed: 20-Jul-2020].

[57] Google, “Google Environmental Report 2018,” 2018.

[58] Urs Hölzle and Google, “Announcing ‘round-the-clock clean energy for cloud,” 14-Sep-2020. [Online]. Available: https://cloud.google.com/blog/topics/inside-google-cloud/announcing-round-the-clock-clean-energy-for-cloud. [Accessed: 20-Sep-2020].

[59] B. Abali, R. J. Eickemeyer, H. Franke, C.-S. Li, and M. A. Taubenblatt, “Disaggregated and optically interconnected memory: when will it be cost effective?,” Mar. 2015.

[60] A. Pagès, J. Perelló, F. Agraz, and S. Spadaro, “Optimal VDC Service Provisioning in Optically Interconnected Disaggregated Data Centers,” *IEEE Communications Letters*, vol. 20, no. 7. pp. 1353–1356, 2016.

[61] K. Katrinis *et al.*, “Rack-scale disaggregated cloud data centers: The dReDBox project vision,” *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. pp. 690–695, 2016.

[62] C.-S. Li, H. Franke, C. Parris, B. Abali, M. Kesavan, and V. Chang, “Composable architecture for rack scale big data computing,” *Futur. Gener. Comput. Syst.*, vol. 67, pp. 180–193, 2017.

[63] R. F. Freitas and W. W. Wilcke, “Storage-class memory: The next storage system technology,” *IBM Journal of Research and Development*, vol. 52, no. 4.5. pp. 439–447, 2008.

[64] J. Zheng and H. T. Mouftah, *Optical WDM Networks: Concepts and Design Principles*, 1st ed. John Wiley & Sons, 2004.

[65] D. Sadot and E. Boimovich, “Tunable optical filters for dense WDM networks,” *IEEE Commun. Mag.*, vol. 36, no. 12, pp. 50–55, Dec. 1998.

[66] F. Hillier and G. Lieberman, *Introduction to operations research*, 9th ed. New York: McGraw-Hill, 2010.

[67] M. Pioro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers, 2004.*, 1st ed. Morgan Kaufmann, 2004.

[68] AMPL Optimization inc, “AMPL.” [Online]. Available: https://ampl.com/products/ampl/. [Accessed: 23-Aug-2020].

[69] AMPL Optimization inc, “CPLEX for AMPL.” [Online]. Available: https://ampl.com/products/solvers/solvers-we-sell/cplex/. [Accessed: 23-Aug-2020].

[70] University of Leeds, “ARC3 – Advanced Research Computing.” [Online]. Available: http://arc.leeds.ac.uk/systems/arc3/. [Accessed: 11-Sep-2018].

[71] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Wiley Publishing, 2007.

[72] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco: Elsevier, 2001.

[73] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge: MIT Press, 2004.

[74] G. Kandiraju, H. Franke, M. D. Williams, M. Steinder, and S. M. Black, “Software defined infrastructures,” *IBM J. Res. Dev.*, vol. 58, no. 2/3, pp. 2:1-2:13, Mar. 2014.

[75] A. Rickman, “The commercialization of silicon photonics,” *Nat Phot.*, vol. 8, no. 8, pp. 579–582, Aug. 2014.

[76] D. Thomson *et al.*, “Roadmap on silicon photonics,” *J. Opt.*, vol. 18, no. 7, p. 073003, Jul. 2016.

[77] Ericsson, “Hyperscale Cloud – reimagining data centers from hardware to applications,” 2016. [Online]. Available: https://www.ericsson.com/assets/local/publications/white-papers/wp-hyperscale-cloud.pdf. [Accessed: 19-May-2017].

[78] P. Costa, H. Ballani, and D. Narayanan, “Rethinking the Network Stack for Rack-scale Computers,” in *6th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, 2014.

[79] S. J. Ben YOO, “The Role of Photonics in Future Computing and Data Centers,” *IEICE Trans. Commun.*, vol. E97.B, no. 7, pp. 1272–1280, 2014.

[80] E. Agrell *et al.*, “Roadmap of optical communications,” *J. Opt.*, vol. 18, no. 6, pp. 0–40, Jun. 2016.

[81] Intel, “Intel® Rack Scale Design.” [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design-overview.html. [Accessed: 30-May-2018].

[82] Gen-Z Consortium, “About – Gen-Z Consortium.” [Online]. Available: https://genzconsortium.org/about/. [Accessed: 11-May-2018].

[83] Hewlett Packard Enterprise, “HPE Composable Infrastructure: Bridging traditional IT and the idea economy | HPE™ United Kingdom.” [Online]. Available: https://www.hpe.com/uk/en/pdfViewer.html?resource=/content/hpe/country/uk/en/resources/solutions/white-paper/4aa5-8813enw&parentPage=/uk/en/what-is/composable-infrastructure. [Accessed: 05-Jan-2018].

[84] Todd Brannon, “Composable Infrastructure, Part 3: ‘What is it??,’” 2015. [Online]. Available: https://blogs.cisco.com/datacenter/composable-infrastructure-part-3-what-is-it. [Accessed: 11-Jan-2018].

[85] Intel, “Intel® Rack Scale Design Architecture Specification,” 2017. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/guide

s/platform-hardware-design-guide-v2-1.pdf. [Accessed: 01-Jun-2018].

[86] Huawei, “High Throughput Computing Data Center Architecture (Thinking of Data Center 3.0).” [Online]. Available: http://www.huawei.com/ilink/en/download/HW_349607&usg=AFQjCNE0m-KD71dxJeRf1cJSkNaJbpNgnw&cad=rja. [Accessed: 13-Sep-2017].

[87] G. Saridis *et al.*, “DORIOS: Demonstration of an All-Optical Distributed CPU, Memory, Storage Intra DCN Interconnect,” *Opt. Fiber Commun. Conf.*, p. W1D.2, 2015.

[88] G. M. Saridis *et al.*, “EVROS: All-optical programmable disaggregated data centre interconnect utilizing hollow-core bandgap fibre,” *2015 European Conference on Optical Communication (ECOC)*. pp. 1–3, 2015.

[89] Y. Yan *et al.*, “All-Optical Programmable Disaggregated Data Centre Network Realized by FPGA-Based Switch and Interface Card,” *J. Light. Technol.*, vol. 34, no. 8, pp. 1925–1932, 2016.

[90] Gen-Z Consortium, “What are the key technical advantages of Gen-Z?” [Online]. Available: https://genzconsortium.org/faqs/. [Accessed: 31-Aug-2018].

[91] I. C. Bulent and A. Paul, “Towards a Composable Computer System,” in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, 2018, pp. 137–147.

[92] G. Zervas, H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra, “Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [Invited],” *IEEE/OSA J. Opt. Commun. Netw.*, vol. 10, no. 2, pp. A270–A285, 2018.

[93] Y. Yan, Y. Shu, G. M. Saridis, B. R. Rofoee, G. Zervas, and D. Simeonidou, “FPGA-based optical programmable switch and interface card for disaggregated OPS/OCS data centre networks,” in *2015 European Conference on Optical Communication (ECOC)*, 2015, pp. 1–3.

[94] H. M. Mohammad Ali, T. E. H. El-Gorashi, A. Q. Lawey, and J. M. H. Elmirghani, “Future Energy Efficient Data Centers With Disaggregated Servers,” *J. Light. Technol.*, vol. 35, no. 24, pp. 5361–5380, 2017.

[95] K. Christodoulopoulos, K. Kontodimas, A. Siokis, K. Yiannopoulos, and E. Varvarigos, “Efficient Bandwidth Allocation in the NEPHELE Optical/Electrical Datacenter Interconnect,” *J. Opt. Commun. Netw.*, vol. 9, no. 12, p. 1145, Dec. 2017.

[96] Ericsson, “Ericsson Hyperscale Datacenter System 8000 Solution Brief,” 2016. [Online]. Available: https://www.ericsson.com/globalassets/pdfs/solution_brief_hds_8000_final.pdf. [Accessed: 19-May-2017].

[97] G. Zervas *et al.*, “Disaggregated compute, memory and network systems: A new era for optical data centre architectures,” in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*,

2017, pp. 1–3.

[98] G. Farias, F. Brasileiro, R. Lopes, M. Carvalho, F. Morais, and D. Turull, “On the Efficiency Gains of Using Disaggregated Hardware to Build Warehouse-Scale Clusters,” in *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, 2017, vol. 2017-Decem, pp. 239–246.

[99] H. M. Mohammad Ali *et al.*, “Energy efficient disaggregated servers for future data centers,” in *2015 20th European Conference on Networks and Optical Communications - (NOC)*, 2015, pp. 1–6.

[100] H. M. M. Ali, A. M. Al-Salim, A. Q. Lawey, T. El-Gorashi, and J. M. H. Elmirghani, “Energy efficient resource provisioning with VM migration heuristic for Disaggregated Server design,” *2016 18th International Conference on Transparent Optical Networks (ICTON)*. pp. 1–5, 2016.

[101] PolettiF. *et al.*, “Towards high-capacity fibre-optic communications at the speed of light in vacuum,” *Nat Phot.*, vol. 7, no. 4, pp. 279–284, Apr. 2013.

[102] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, “A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems,” *Adv. Comput.*, vol. 82, pp. 47–111, Jan. 2011.

[103] Cisco, “Cisco Nexus 3132C-Z Switches Data Sheet - Cisco.” [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-3000-series-switches/datasheet-c78-740895.html. [Accessed: 22-Oct-2018].

[104] Finisar Corporation, “WaveShaper-16000S-Product-Brief.” [Online]. Available: www.finisar.com/instruments. [Accessed: 07-Sep-2018].

[105] Polatis, “Polatis - Series 6000 single mode all optical low loss switch 4x4 to 192x192 ports.” [Online]. Available: http://www.polatis.com/polatis-series-6000-optical-matrix-switch-192x192-sdn-enabled-industry-leading-performace-lowest-loss-switches.asp. [Accessed: 07-Sep-2018].

[106] P. X. Gao *et al.*, “Network Requirements for Resource Disaggregation,” in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016, pp. 249–264.

[107] R. Lin *et al.*, “Real-time 100 Gbps/λ/core NRZ and EDB IM/DD transmission over multicore fiber for intra-datacenter communication networks,” *Opt. Express*, vol. 26, no. 8, p. 10519, Apr. 2018.

[108] X. Pang *et al.*, “200 Gbps/Lane IM/DD Technologies for Short Reach Optical Interconnects,” *J. Light. Technol.*, vol. 38, no. 2, pp. 492–503, Jan. 2020.

[109] Q. Cheng, A. Wonfor, J. L. Wei, R. V Penty, and I. H. White, “Low-energy, high-performance lossless 8×8 SOA switch,” in *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, 2015, pp. 1–3.

[110] H. A. Alharbi, T. E. H. Elgorashi, and J. M. H. Elmirghani, “Energy

efficient virtual machines placement over cloud-fog network architecture," *IEEE Access*, vol. 8, pp. 94697–94718, 2020.

[111] I. S. B. M. Isa, T. E. H. El-Gorashi, M. O. I. Musa, and J. M. H. Elmirghani, "Energy efficient fog-based healthcare monitoring infrastructure," *IEEE Access*, vol. 8, pp. 197828–197852, 2020.

[112] B. A. Yosuf, M. Musa, T. Elgorashi, and J. Elmirghani, "Energy Efficient Distributed Processing for IoT," *IEEE Access*, vol. 8, pp. 161080–161108, 2020.

[113] Cisco, "Application hosting on Catalyst 9000 series of switches - Cisco DevNet." [Online]. Available: https://developer.cisco.com/docs/app-hosting/#!application-hosting-in-the-enterprise/what-is-application-hosting. [Accessed: 21-Apr-2020].

[114] K. David, J. Elmirghani, H. Haas, and X. H. You, "Defining 6G: Challenges and Opportunities [From the Guest Editors]," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3. Institute of Electrical and Electronics Engineers Inc., pp. 14–16, 01-Sep-2019.

[115] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, 2018.

[116] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 3909–3914.

[117] A. N. Al-Quzweeni, A. Q. Lawey, T. E. H. Elgorashi, and J. M. H. Elmirghani, "Optimized Energy Aware 5G Network Function Virtualization," *IEEE Access*, vol. 7, pp. 44939–44958, 2019.

[118] N. Chen, Y. Yang, T. Zhang, M. Zhou, X. Luo, and J. K. Zao, "Fog as a Service Technology," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 95–101, 2018.

[119] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, 2016.

[120] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things," in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, 2013, pp. 15–20.

[121] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog Computing: Towards Minimizing Delay in the Internet of Things," in *2017 IEEE International Conference on Edge Computing (EDGE)*, 2017, pp. 17–24.

[122] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, 2016.

[123] Juniper Networks, "NFX SERIES NETWORK SERVICES

PLATFORM.” [Online]. Available: https://www.juniper.net/assets/uk/en/local/pdf/datasheets/1000563-en.pdf. [Accessed: 28-Mar-2019].

[124] Juniper Networks, “MX Series 5G Universal Routing Platforms Product Description.” [Online]. Available: https://www.juniper.net/assets/uk/en/local/pdf/datasheets/1000597-en.pdf. [Accessed: 28-Mar-2019].

[125] Cisco, “Cisco ME 4600 Series Optical Network Terminal Data Sheet - Cisco,” 2017. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/me-4600-series-multiservice-optical-access-platform/datasheet-c78-730446.html. [Accessed: 28-Mar-2019].