Time-varying Resilient Virtual Networking Mapping for Multi-location Cloud Data Centers

Ting WANG

A thesis

in The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

January 2016

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Ting WANG

Entitled: Time-varying Resilient Virtual Networking Mapping for Multi-location Cloud Data Centers

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. E.J.Doedel

_____ Examiner

Dr. Lata Narayanan

_____ Examiner

Dr. Thomas Fevens

_____ Supervisor

Dr. B.Jaumard

Approved By _____

Chair of Department or Graduate Program Director

_____ 2016 _____ _____

Name

Faculty of Engineering and Computer Science

Abstract

In the currently dominant cloud computing paradigm, applications are being served in data centers (DCs), which are connected to high capacity optical networks. For bandwidth and consequently cost efficiency reasons, in both DC and optical network domains, virtualization of the physical hardware is exploited. In a DC, it means that multiple so-called virtual machines (VMs) are being hosted on the same physical server. Similarly, the network is partitioned into separate virtual networks, thus providing isolation between distinct virtual network operators (VNOs). Thus, the problem of virtual network mapping arises: how to decide which physical resources to allocate for a particular virtual network? In this thesis, we study that problem in the context of cloud computing with multiple DC sites. This introduces additional flexibility, due to the anycast routing principle: we have the freedom to decide at what particular DC location to serve a particular application. We can exploit this choice to minimize the required resources when solving the virtual network mapping problem.

This thesis solves a resilient virtual network mapping problem that optimally decides on the mapping of both network and data center resources, considering time-varying traffic conditions and protecting against possible failures of both network and DC resources. We consider the so-called VNO resilience scheme: rerouting under failure conditions is provided in the virtual network layer. To minimize physical resource capacity requirements, we allow reuse of both network and DC resources: we can reuse the same resources for the rerouting under failure scenarios that are assumed not to occur simultaneously. Since we also protect against DC failures, we allocate backup DC resources, and account for synchronization between primary and backup DCs. To deal with the time variations in the volume and geographical pattern of the application traffic, we investigate the potential benefits (in terms

of overall bandwidth requirements) of reconfiguring the virtual network mapping from one time period to the next. We provide models with good scalability, and investigate different scenarios to check whether it is worth to change routing for service requirement between time periods. The results come up with our experiments show that the benefits for rerouting is very limited.

Keywords: Cloud Computing, Optical Networks, Virtualization, Anycast, VNO resilience

Acknowledgments

I am grateful to my supervisor, Dr. Brigitte Jaumard for her encouraging, personal guidance and her efforts to explain things, have provided a good basis for the present thesis. It's my fortune to have her as my supervisor. I particularly appreciated that despite her enormous workload she always made herself available to me and took the time to revise my thesis and papers for publications.

Next I would like to thank all the faculty members and staff of the Computer Science Department. I am grateful to the Faculty of Engineering and Computer Science, Concordia University for supporting in part of this thesis work. Also I would like to express my deep appreciation to the members of the Committee: Professors Narayanan, Harutyunyan for their valuable feedback on my thesis. Their comments, questions and suggestions have been very useful to improve my work.

I wish to express my warm and sincere thanks to Professor Chris Develder, who gave various insights of the current standards and market needs.

Lastly, I dedicate this thesis to my parents, who always encourage me and offer continuous moral support by wishing me well in my career. Their boundless love and dedication are always the inspiration throughout my life.

Contents

List of Tables

List of Figures

Abbreviations

API Application Programming Interface

BLP Bite-Level Parallelism

CG Column Generation

COW Cluster of Workstations

CRM Customer Relationship Management

DC Data Center

DSM Distributed Shared Memory

FID Failure-independent

FIFR Failure Inferencing based Fast Rerouting

FTTH Fiber-to-the-Home

GPU Graphics Processor Unit

HPC High-performance Computing

HP Homogeneous Poisson

IaaS Infrastructure as a Service

ILP Integer Linear Program

JNI Java Native Interface

JVM Java Virtual Machine

LP Linear Program

MILP Mixed Integer Linear Program

MIMD Multiple Instruction, Multiple Data Stream

MMPP Markov Modulated Poisson Process

MPI Message Passing Interface

MPJ Message Passing in Java

MPLS Multiprotocol Label Switching

MPP Massively Parallel Processor

MP Master Problem

OFDM Orthogonal Frequency Division Multiplexing

OPSIX Portable Operating System Interface

OTN Optical Transport Network

P2P Peer-to-peer

PaaS Platform as a Service

PE Pareto-Exponential

PIP Physical Infrastructure Provider

PP Pricing Problem

PVP Parallel Vector Processor

QoS Quality of Services

RMP Restricted Master Problem

RVNM Resilient Virtual Networking Mapping

SaaS Software as a Service

SDM Shared Distributed Memory

SIMD Single Instruction Multiple Data Stream

SISD Single Instruction, Single Data Stream

SMP Symmetric Multiprocessor

SRLG Shared Risk Link Group

VM Virtual Machine

VNet Virtual Network

VNO Virtual Network Operator

WDM Wavelength Division Multiplexed

Chapter 1

Introduction

This chapter begins by laying out the general background of the thesis in Section 1.1 and the introduction of the research project, called the Resilient Virtual Networking Mapping (RVNM) problem, in Section 1.2. Finally, key contributions and organization of this thesis are described in Section 1.3 and 1.4, respectively.

1.1 General Background

In the past few decades, there is a sharp increase in Internet usage. According to Internet World Stats [4], today's Internet has approximately 3 billion users. National Science Foundation gives a prediction that the number of Internet users will increase to nearly 5 billion by 2020 [1].

The network bandwidth increases rapidly to support the high bandwidth demand of the entertaining applications, and the fact is relatively well-known, due to the improvement in the capacity and affordability of processors, memory, disks, etc.

In order to satisfy the increasing global bandwidth requirement, optical networks can definitely be considered due to the advantages of transmission speed, expansion capacity and stability. Optical fiber has already been deployed in the backbone and in the metropolitan networks. The current trend is to let it penetrate into the access network domain and achieve FTTH (fiber to the home) ultimately.

However, the multi-provider nature of the Internet and the consensus requirements bring

a big challenge for architecture alterations such as incremental updates and new network technologies deployment. So network virtualization has been considered as a key attribute for the future network.

The basic idea behind network virtualization is to allow multiple coexisting heterogeneous network architectures by splitting the roles of the traditional Internet service providers into two independent entities. One is Physical Infrastructure Providers (PIPs), who create and manage the physical infrastructure. Another one is Virtual Network Operators (VNOs), who create virtual networks (VNets) by aggregating resources from multiple PIPs and over end-to-end services [13].

Moreover, we also need to introduce the concept of cloud computing. Cloud computing is a computing term or metaphor that evolved in the late 2000s, based on distributed computing, parallel computing and grid computing which will be introduced in next chapter. It is an Internet-based super-computing model that has tens of computers and servers connected with each other in remote data centers (DCs). The basic principle of cloud computing is to distribute computing on a large number of distributed computers rather than the local computer or a single remote server, so that users can get the required resources (hardware, platform, software) through the network. The network providing resources is called "Cloud". From the users' point of view, the resources in a "Cloud" is "infinitely" expandable, and can be readily available, on-demand delivered [39].

With the advent of the era of big data, cloud storage, one extension of cloud computing also appears in public view. It is a kind of online storage mode, that is, the data stored by a third party which is usually hosted in multiple virtual servers rather than on a dedicated server. Data files can be stored in different storage nodes in a cloud storage system, and those data files can be accessed without geographical restrictions [33]. When the system upgrade or a failure happens, user's requirement could be satisfied by guiding the I/O instructions to another storage server who has same data files, and after b the original storage server recovers, the file will then migrate back. This is one of the most important context of our model. A high level architecture of cloud storage is illustrated in Fig. 1.1.

Our RVNM project focuses on IP-over-WDM networks, and our main goal is to optically solve a mapping problem that minimizes the bandwidth resources considering time-varying

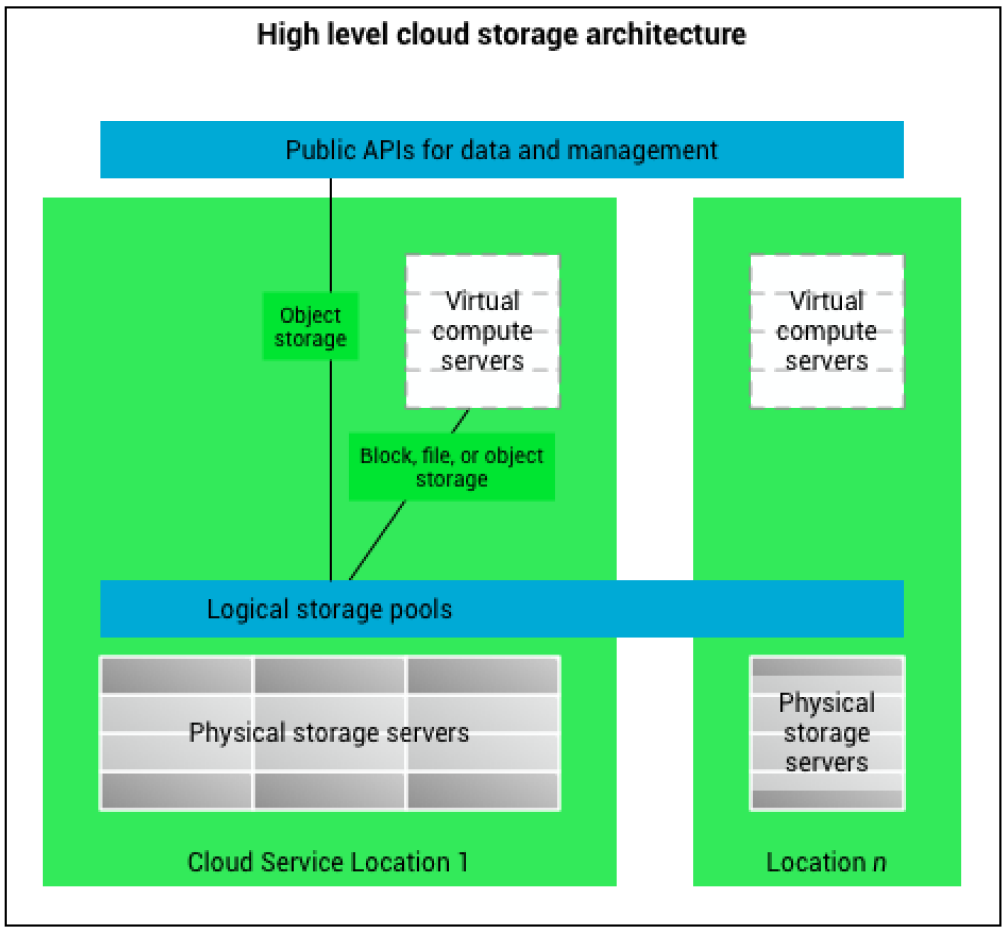

Fig. 1.1: A High Level Architecture of Cloud Storage (taken from [2]).

traffic conditions and protecting against possible failures of both network and DC resources. Here, mapping refers to how to determine a VNet topology on top of physical infrastructure. In context of cloud computing and cloud storage, we have multi-location DC resources, and do not really care about the location of each DC. Therefore, our project need to decide on the mapping of both network and the DC resources.

1.2 Introduction and Motivation of Thesis

Cloud computing is really attractive today as a kind of on-demand computing. Many big companies are already provided services based on clouds with pay-as-you-go pricing. Not only the industries are interested in involving this technique, a lot of individual Internet users are start to using the services provided by clouds. One of the services widely used by individuals

is cloud storage, for example Google Drive provided by Google. Since the requirements for quality of services (QoSs) is keeping increasing with the improvement of network technologies, the survivability of optical networks, to support cloud services (distributed over multiple locations), becomes a critical concern.

		Time to repair	
Cause	Events	Avg	Med
Hardware	20%	95 m	5 m
Power	6%	93 m	18 m
External	15%	61 m	4.6 m
Software	32%	10 m	4 m
Configuration	9%	5 m	1 m
Other	12%	46 m	6 m
Unknown	5%	52 m	6 m

Fig. 1.2: Major Causes of Failure According to Administrator Announcements (taken from [57]).

Cause	Notices	Scheduled	Impacting
Hardware	25%	65%	71%
Power	20%	4%	99%
External	15%	29%	95%
Software	12%	84%	99%
Other	12%	69%	82%
Configuration	8%	91%	45%
Unknown	7%	0%	99%

Fig. 1.3: Breakdown of Administrator Notices by Failure Cause (taken from [57]).

An end-to-end network is consisted by many network elements, and each of them could fail at any time. Since there are many reasons for the failures such as natural disasters, human errors, power off and so on, network failure could not be fully avoided. Fig. 1.2 and Fig. 1.3

shows the major causes of failure and impact collected and analyzed based on CENIC (the Corporation for Education Network Initiatives in California) network and the log information from the network extending from late 2004 to the end of 2009 [57]. As we mainly focus on VNets, the failure causes can be summarized as failures on links and data centers. In this thesis, we develop models to protect against single failure (assuming there is only one failure happen on the network at one time) by providing a working path, a backup path as well as a working (primary) data center and a backup data center for each service request. In cloud context, users do not care very much about the exact location of data centers, and service providers can exploit anycast routing: to serve a new request, they basically have the freedom to pick any of the available data centers. As a result, this anycast principle can be exploited for resiliency purposes: if either the server infrastructure (in the DC), or the (optical) network is affected by a failure, the data delivery on a working path will be switched to a backup path which refers to a different DC location. In order to eliminate the disruption of data delivery when switching paths, a synchronization path to synchronize the data between working and backup paths is also generated for each request.

As we mentioned at beginning, the network bandwidth increases rapidly and there is no doubt it will keep increasing in the future. Therefore, how to determine routing for service requests in order to lower bandwidth cost is worth to be studied. Since we are taking care of single-failure protection, a backup path for a request will only be used when there is a failure happens on the working path of this request. That means, only the backup paths for requests which share same link/node on their working paths has the possibility to be used at same time. Therefore, it is possible to share backup paths for those requests which do not share link/node for working paths. While shared protection schemes allow significant bandwidth saving, additional saving can be achieved by re-provisioning (rerouting) paths if the traffic is highly time-varying.

This topic has been investigated in the past, but not thoroughly. To our knowledge, we are the first who considered resilient multi-period anycast traffic routing. This motivated us to formulate the optimization problem to find the best routing going from one period to the next.

1.3 Thesis Contributions

In this thesis, we solved a resilient virtual network mapping problem that optimally decides on the mapping of both network and data center resources, considering time-varying traffic conditions and protecting against possible failures of both network and DC resources. We proposed a global optimization model, considering optimization of the routing over a set of multiple consecutive time periods.

Besides, we studied the interest of re-provisioning the working and the backup paths in the context of resilient anycast routing traffic in cloud computing, assuming time-varying traffic, where the path provisioning can be updated periodically. We consider a multi-time periods approach, where the traffic requests change from one time period to the next, and investigate the usefulness of reconfiguration the traffic routes when a new time slot starts. Such reconfiguration may involve changing working and/or backup paths for (some of) the traffic flows. Since changing the working path of ongoing traffic might be too disruptive (or unacceptable for some time-critical, high quality of services(QoS)), we also investigate the potential benefit (in terms of overall reduced link bandwidth occupancy) of only modifying the backup paths.

In our pervious work, we provided a single-time period model that considers traffic variation between multiple time periods. The model is designed for solving the problem for a single time period at one step: given the virtual network mapping for time t, we determine the (possibly changed) mapping for t + 1. In this thesis, the models for truly multi-period traffic is developed, and they could determine the routing for all time periods at once. We propose several column generation models, but only implemented the most scalable one with parallel computing technology.

Publications:

[1] Bui, Minh, Ting Wang, Brigitte Jaumard, Deep Medhi, and Chris Develder. "Time-varying resilient virtual network mapping for multi-location cloud data centers." In IEEE 16th International Conference on *Transparent Optical Networks (ICTON), 2014*, pp. 1-8.

Submitted:

[2] Ting Wang, Brigitte Jaumard, and Chris Develder. "A Scalable Model for Multi-period

Virtual Network Mapping for Resilient Multi-Site Data Centers." In IEEE International Conference on *Advanced Networks and Telecommunications Systems (ANTS), 2015.*

To be shortly submitted:

[3] Ting Wang, Brigitte Jaumard, and Chris Develder. "A Scalable Model for Multi-period Virtual Network Mapping for Resilient Multi-Site Data Centers." *Journal of Optical Communications and Networking.*

1.4 Organization of Thesis

The thesis is organized as follows. In Chapter 2, it begins by laying out the technical background, then we present a literature review on the previously published studies related to this resilient virtual network mapping (RVNM) problem. In Chapter 3, we give a statement of our RVNM problem, explain our mapping scheme in details and highlight the three different rerouting scenarios we consider. In Chapter 4, we present four different mathematical models for finding the routes that minimize the bandwidth requirements to serve time-varying cloud traffic, and analysis the advantages and drawbacks of each model. In Chapter 5, we introduce the solution process of RVNM problem by describing the implemented parallel strategy. The numerical results is given in Chapter 6. We summarize our conclusions and future work in Chapter 7.

Chapter 2

Background and Literature Review

In this chapter, an overview of technical background is provided in Section 2.1. Then, we will present a literature review regarding to rerouting in networks and resilient virtual topologies mappiong in Section 2.2.

2.1 Background

In this section we will review the basic technique concepts of this thesis including the grids and clouds, anycast routing, column generation and parallel computing.

2.1.1 Grids vs. Clouds

The term "Grid "comes from the word"power grid"which is familiar by public. A grid is indeed similar to a power grid in many aspects. When people wash their clothes by using a washing machine, the only thing they care about is that when the clothes can be cleaned, but not where the power comes from. Similar to grids, users do not need to consider about where the resources (computational, storage and networking) are, and they only take care of what kind of services they are expecting. Grids can be simply understand as a from of distributed system that interconnect the whole network as a "super virtual computer", in which computing resources are not administered centrally, open standards are used, and notrivial quality of service is achieved [24].

Foster *et al.* [25] presented the layered grid architecture which contains five layers and is illustrated in Fig. 2.1.

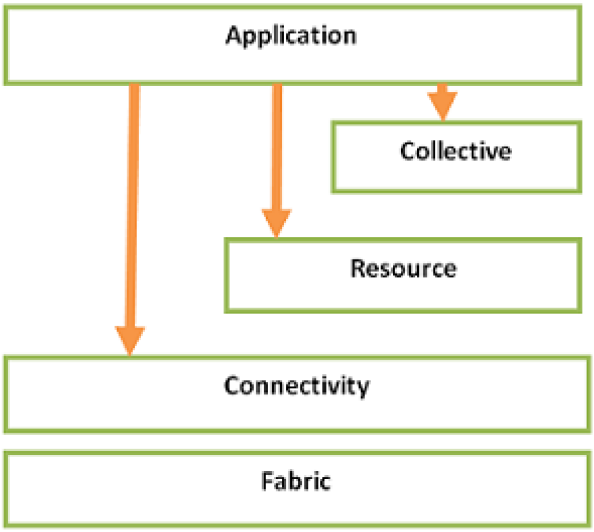

Fig. 2.1: The Five Layered Grid Architecture (taken from [47])

- Fabric: Interfaces to local control. It is composed of physical or logical elements to provide shared resources to upper layer. The commonly used physical resources include computational resources, storage systems, catalogs, network resources. The logic resources are distributed file systems, distributed computing pool, computer systems and so on. The functions implemented at the fabric level can be affected by the demand of higher levels. Guarantee of the quality of service (QoS) for resource discovery and management are the basic functions of this layer.

- Connectivity: Communicating easily and securely. The core communication and authentication protocols are defined on connectivity layer. It also take the responsibility of data exchange between resources, as well as giving authentication and security solutions.

- Resource: Sharing single resources. The resource layer builds on the communication and authentication protocol which is defined on connectivity layer. It implement the protocols for the secure negotiation, resource initiation, monitoring and statistics. The individual resources are accessed and controlled by calling fabric-layer functions.

- Collective: Coordinating multiple resources. Collective layer collect the resources submitted by resource layer which could be invoked by applications in order to achieve resource sharing.

- Applications. Applications access to the required services by invoking the application programming interface (API) defined by each layer. The resources of the grid was used by these services to complete tasks.

This five-layered architecture h focused primarily on qualitative description rather than specific defined protocol.

According to Li *et al.* [34] and Travostino *et al.* [56], we summarize four advantages of grids which are described as follows:

- Resource sharing . Grids can provide resource sharing and eliminate information silos,in order to achieve the interconnection of applications. Different from the traditional computer networks, grid can provide communications on the application layer.

- Collaborative working. Working collaborative is the second feature of grids. A lot of grid nodes can work together on a single project.

- Decentralized management and control. Grids are using standard, open, general-purpose protocols and interfaces, so that resource provisioning, utilization and reconfiguration can be allowed without any other authorities include the centralized management.

- Dynamic integration and scalability. Grids enable services and resources to be integrated and continually changed dynamically.

Nowadays, all kinds of grids are being applied in an expanding scope of applications and services, especially in Internet and web technology. A grid netwok is a computer network that includes a number of devices such as supercomputers, storage elements and file systems connected in a grid topology. In grid networks, distributed resources, computing or storage

elements as well as scientific instruments are incorporated with a communications capability to support computing-intensive and data-intensive applications [52].

Inspired by the success of the grid paradigm in scientific circles, the cloud computing ideas arose in the 2000s, building on the seminal idea of "computation provided as a public utility" (as suggested back in 1961 by John McCarthy) [18]. Clouds inherit a lot of advantages of grids, and have some differences. The relationship of clouds, grids and some other technologies are shown in Fig. 2.2. I gave a simple introduction of cloud computing in Section 1.1. Here, we will talk about some of the differences between grids and clouds.

- Business Model. Cloud computing refers to the on-demand delivery of IT resources and applications via the Internet with pay-as-you-go pricing. Whereases, the business model for grids (at least that found in academia or government labs) is project-oriented in which the users or community only have certain number of service units (i.e., CPU hours) they can spend [26].

- Infrastructure. Cloud applications typically run in large data centers, as opposed to high-performance computing (HPC) infrastructure for many grid applications.

- Monitoring. Monitoring in clouds is quite challenging whereas grids apply a different trust model where users, via identity delegation, can access and browse resources at various sites that contain resources. In grids, these resources are typically not that much abstracted or virtualized compared to Clouds. Because the user can not deploy their own monitoring infrastructure and the returned information may not provide enough details to figure out the resource status.

- Virtualization. It is a key difference between clouds and grids. It enables migration to other servers, both for performance and resilience against failures.

Cloud computing is one of the most promising technology developed from grid computing which has already been provided by some strong industries, such as Google (Google App Engine, a cloud platform supported by Google) and Amazon (Amazon Web Services offer cloud computing services). Clouds are likely to provide everything as a service which is classified into three levels. "Infrastructure as a Service (IaaS) provisions hardware, software,

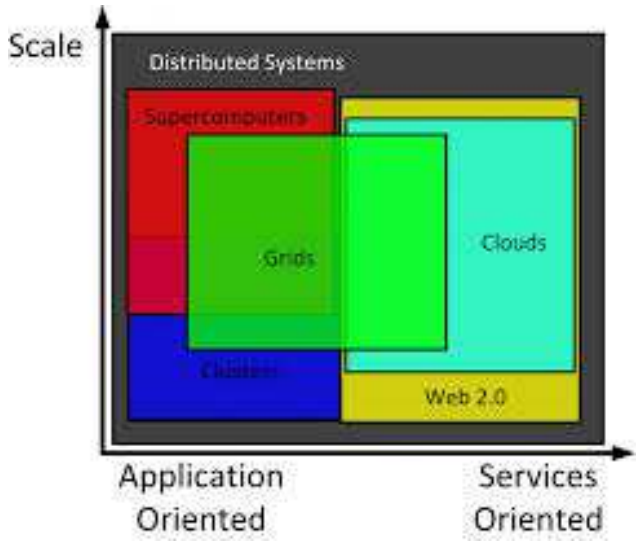

Fig. 2.2: Grids and Clouds Overview (taken from [26])

and equipment (mostly at the unified resource layer, but can also include part of the fabric layer) to deliver software application environments with a resource usage-based pricing model. " [26] Infrastructure can scale up and down dynamically based on application resource needs. Platform as a Service (PaaS) offers a integrated development environment to build, test, and deploy applications. Generally, developers will need to accept some restrictions on the type of software they can write in exchange for built-in application scalability. Software as a Service (SaaS) delivers special-purpose software that is remotely accessible by consumers through the Internet with a usage-based pricing model. SaaS model can remove the maintenance cost as well as saving the budget on buying software and hardware.

The main goal of our project is to give an efficient routing and scheduling strategy for traffic in a context of cloud with the resources optimal principle. However, it's also adapted to Grid networks.

2.1.2 Anycast Routing

In an optical grid network and the cloud context, resouces are stored in multiple locations and users do not care about which location they are getting connected to. Therefore there are a group of potential destinations, and a certain source-destination pair does not exist anymore [19]. Moreover, in order to meet the demand of geographically wide-spread users,

the efficiency of data delivery and high user-perceived quality of service should be remained at the acceptable and competitive level. [8]. These two points address us to use anycast as our routing method.

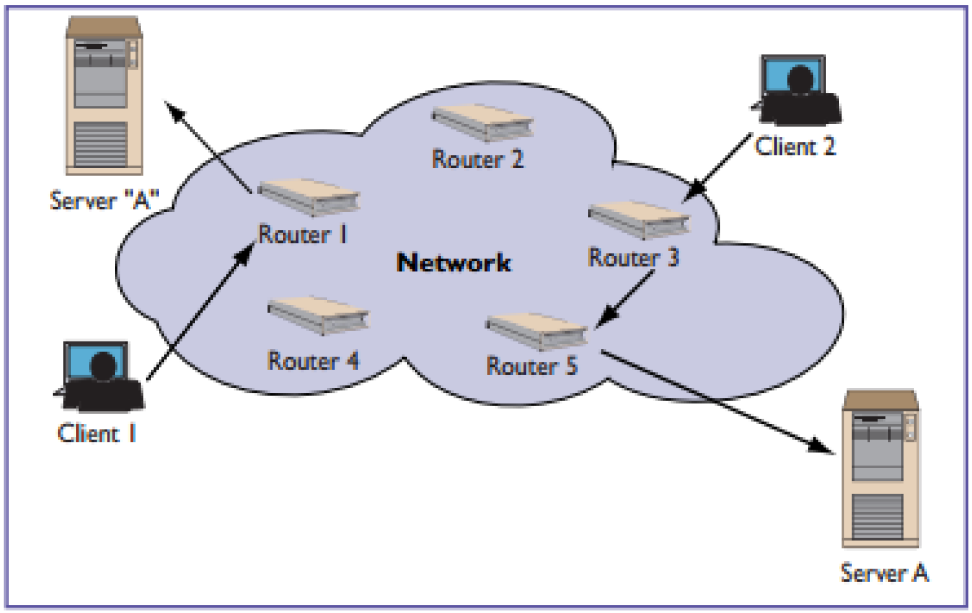

Fig. 2.3: An Example of Grid Network (taken from [40])

Anycast is originally defined as an address method that provides a stateless best effort delivery of an anycast datagram to at least one host, and preferably only on host, which serves the anycast address [46]. In this definition, a client send packets to any one of a server group who is offering a particular service or application, but will not care about which one has been sent to.

Fig. 2.3 illustrates the anycast notion within the Network-layer (or IP). As Fig. 2.3 shows, all servers in an anycast group own the same IP address. When clients try to get access of the services, anycast. packets flow will choose from a client to the nearest destination server identified by anycast address A. Then the network treats an anycast address (the anycast address stands for one specified server in the servers group) as a specific host address and the IP address is assigned to the selected anycast address .

Anycast is attrative when considering services provided in grid/cloud environment, Louwersheimer [36] summarized several reasons, and part of them are related to IP anycast are listed as follows:

- Reduced use of router and link resources. Standard IP routing will deliver packets over the shortest path to closest available server.
- Simplified configuration. A client only needs to be configured with a single anycast address that identifies one of a group of possible servers offering a particular service or application.
- Network resiliency. If a server in the anycast group goes away, the network will deliver packets to the next closest anycast server. The service will become more reliable.
- Load balancing. Anycast servers distributed over the network topology will have the effect of balancing the traffic load from many clients.

There are two anycast schemes, according to Metz [40], one is the network-layer (or IP) anycast, and another one is the application-layer anycast. The network-layer anycast is solely based on network topology. If the destination host is selected due to the factors such as the fewest router hops or lowest cost (there may be different cost between router links), then we attribute it to network-layer anycast. The metric related to application characteristics such as response time, capacity and active connections is considered by application-layer anycast. The external entity that application-layer anycasting depends on could help clients to determine the best destination host to contact which is can be guaranteed by network-layer anycast.

We use network-layer anycast as the address method in our RVNM project, however, because of the inherent benefits listed above and the complications and scalability problems exhibited by application-layer anycast [32].

2.1.3 Column Generation

In our project, we use CPLEX which is a powerful optimization software package developed by IBM for linear programming (LP) as our solver tool. Though, there are some other well-known softwares such as Gurobi, GNU Linear Programming Kit and LP_Slove.

Column generation (CG) is an efficient technique for solving large linear programs. The very basic idea of using column generation to solve linear programs was believed to be first

proposed by Ford and Fulkerson [27], and the resulting algorithm is what we call Dantzig-Wolfe decomposition [15]. The idea of CG can be generalized to yield an algorithm for solving any LP by partitioning the problem into two problems: the master problem (MP) and the pricing problem (PP) [44]. The MP is the original problem, and usually looks like follows (which c and a is matrix, b is a given value):

$$\begin{aligned} z := \min & \sum_{j \in J} c_j \lambda_j \\ \text{subject to} \quad & \sum_{j \in J} \mathbf{a}_j \lambda_j \leq \mathbf{b}, \\ & \lambda_j \leq 0, \quad j \in J. \end{aligned} \tag{2.1}$$

In practice, the $|J|$ is usually huge, and because the memory of computer is limit, it can not solve such a big model. Therefore, we need to use column generation approach. MP is generally solved with a reasonably small subset $J' \in J$. We call such a problem restricted master problem (RMP) [37]. The job of PP is looking for a new column that could help RMP to get a better solution by using the optimal dual variables provided by the RMP. That is, in the pricing step, we need to find a solution with a negative reduced cost.

Most of linear problems are involved with some integers, then integrated to integer linear program (ILP) problems or mixed integer linear program (MILP) problems. In these cases, as shown in Fig. 2.4, we solve the problem in two steps. First, the problem is solved as LP problem, and it's called LP relaxation. The second step is to get an ILP solution by using the results from the LP relaxation. Therefore, the solution we get is not the real optimal one, but as close as possible to the real optimal one. In other words, the less the gap between LP and ILP/MILP solution, the better the accuracy is. In order to do this, branch-and-cut [45] algorithm is used to derive an integer solution from an optimal LP solution, and brand-and-price [7] algorithm is used to improve the gap between to solutions.

2.1.4 Parallel Computing (MPI, MPJ)

Parallel computing is a form of computation that solve problems by using multiple computational resources simultaneously. It is an effective method to improve the computing speed

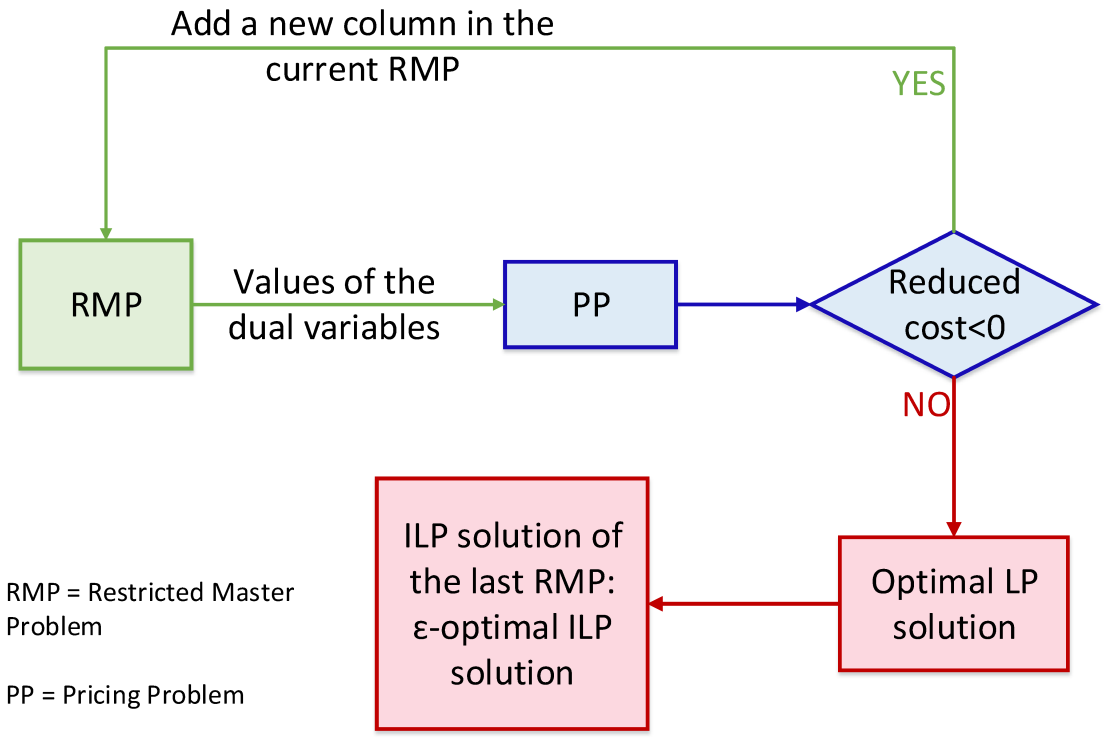

Fig. 2.4: Flowchart of Column Generation

and capacity of computer systems. The basic idea of parallel computing is to let multiple processors to solve the same problem synergistically. In other words, the large problem that need to be solved can be divided into smaller independent ones, and at same time each of them is then solved by a processor which is standalone. Parallel computing system can be either a super computer with multiple specially designed processors or a cluster of connected computers which can work independently. There are several different forms of parallel computing: bit-level, instruction-level, and data, task parallelism [22].

- Bit-level (BLP). Bit-level parallelism is based on increasing processor word size, in order to increase the efficiency of computing operations by decreasing the number of instructions.

- Instruction-level parallelism. In a computer program, a stream of instructions can be carried out at same time. It's also a measure that how many instruct can be executed simultaneously.

- Data parallelism. Data parallelism is a form of parallel computing where same calculation is performed on the same or different sets of data.

- Task parallelism. It's also called function parallelism or control parallelism. It focuses on distributing tasks (concretely performed by processes or threads) across different parallel computing nodes. Just contrasts to data parallelism.

Parallel computing can be classified as parallel-in-time and parallel-in-space. Parallel-in-time is a kind of pipelining technique. Two or more operations will be executed at same time, which could improve computing performance greatly. Parallel-in-space refers to computing carried out by multiple processors, which is connecting two or more processors by network to let them working on different part of one task. By doing this, some large computing problem that can not be solved by individual computer can be handled.

Parallel computing science is mainly focus on parallel-in-space. It led to the creation of two types of parallel machine. According to Flynn's taxonomy [23] [21], they are Multiple Instruction, multiple Data stream (MIMD) and Single Instruction, Multiple Data streams (SIMD). The sequential computers which exploits no parallelism in either the instruction or data streams is also called Single Instruction, Single Data stream (SISD). SIMD refers to a computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized, e.g. an array processor or graphics processor unit (GPU). And MIMD refers to the machine that could support multiple autonomous processors simultaneously carried out different instructions on different data. MIMD machines can be classified into parallel vector processor (PVP), symmetric multiprocessor (SMP), massively parallel processor (MPP), cluster of workstations (COW) and distributed shared memory (DSM) processor.

Concurrent programming languages, libraries, APIs, and parallel programming models have been created for programming parallel computers. These can generally be divided into classes based on the assumptions they make about the underlying memory architecture–

shared memory, distributed memory, or shared distributed memory (SDM). Shared memory programming languages communicate by manipulating shared memory variables. Distributed memory uses message passing. POSIX Threads which is aPortable Operating System Interface (OPSIX) standard for threads and OpenMP are two of most widely used shared memory APIs, whereas Message Passing Interface (MPI) is the most widely used message-passing system API. One concept used in programming parallel programs is the future concept, where one part of a program promises to deliver a required datum to another part of a program at some future time.

2.1.4.1 Message Passing Interface (MPI)

Nowadays, due to the clusters' scalability, flexibility and acceptable ratio performance/cost, it has an important presence in High-Performance Computing (HPC). Currently, multi-core clusters are the most popular option for the deployment of HPC infrastructures. These systems are usually programmed with native languages using message passing libraries, especially Message Passing Interface (MPI) [4], which are targeted to distributed memory systems.

The message–passing is the most widely used parallel programming model as it is portable, scalable and usually provides good performance. It is the preferred choice for parallel programming distributed memory systems such as clusters, which provide higher scalability and performance than shared memory systems. Regarding native languages, Message Passing Interface (MPI) is the standard interface for message-passing libraries [38].

Message Passing Interface (MPI) is a standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computers. The standard defines the syntax and semantics of a core of library routines useful to a wide range of users writing portable message passing programs in different computer programming languages such as Fortran, C, C++ and Java. There are several well-tested and efficient implementations of MPI, including some that are free or in the public domain [54]. These fostered the development of a parallel software industry, and encouraged development of portable and scalable of large-scale parallel applications.

The MPI interface is meant to provide essential virtual topology, synchronization, and

communication functionality between a set of processes (that have been mapped to nodes/servers/ computer instances) in a language-independent way, with language-specific syntax (bindings), plus a few language-specific features. MPI programs always work with processes, but programmers commonly refer to the processes as processors. Typically, for maximum performance, each CPU (or core in a multi-core machine) will be assigned just a single process. This assignment happens at runtime through the agent that starts the MPI program, normally called mpirun or mpiexec.

2.1.4.2 Message Passing in Java (MPJ)

Java has became a leading programming language, especially for distributed programming, and is an emerging option for High-Performance Computing (HPC). The increasing interest on Java for parallel computing is based on its appealing characteristics: built-in networking and multithreading support, object orientation, platform independence, portability, and security [54]. Currently, the hybrid architecture (shared/distributed memory) of the multi-core systems demands the use of hybrid programming approaches, such as the use of MPI+OpenMP, in order to take advantage of the available processing power. An interesting alternative is the use of Java for parallel programming multi-core systems. In fact, the Java built-in networking and multithreading support makes this language especially suitable for this task.

Java can take full advantage of hybrid architectures using intra-process communication in shared memory and relying on efficient inter-node communication. Moreover, Java can handle the increasing availability of computing resources thanks to its portability and the use of scalable communication middleware. Therefore, as scalability is a key factor to confront new challenges in parallel computing, Java message-passing libraries providing such feature through the use of efficient nonblocking communications and high-speed networks support. There have been several implementations of Java message-passing libraries [53]. Most of them have developed their own MPI-like binding for the Java language. The two main proposed APIs are the mpiJava API and MPJ API [12], whose main differences lay on naming conventions of variables and methods.

The mpiJava [5] library consists of a collection of wrapper classes that called a native

MPI implementation (e.g., MPICH or OpenMPI) through Java Native Interface (JNI). This wrapper based approach provides efficient communication relying on native libraries, adding a reduced JNI overhead. However, mpiJava currently only supports some combinations of JVMs(Java Virtual Machines) and MPI libraries, as wrapping a wide number of functions and heterogeneous runtime environments entails an important maintaining effort. Additionally, this native MPI implementation presents instability problems, derived from the native code wrapping, and it is not thread-safe, being unable to take advantage of multi-core systems through multithreading. The main mpiJava drawbacks are solved with the use of pure Java (100% Java) message-passing libraries, that implement the whole messaging system in Java. However, these implementations are usually less efficient than mpiJava. MPJ Express [6] is a thread-safe and pure MPJ library that implements the mpiJava API.

We use Java language to implement our mathematical model. Though most people will choose C or C++ as their programming language when they implement their project related to HPC (because of the running speed of memory usage of these languages). As an objective oriented programming language, Java is much more friendly to programmers, and the automatic garbage collection could saves a lot of time and helps to avoid some memory issues to the junior programmer who are not so familiar with memory management. As a consequence, MPJ is our best choose to implement a parallel program.

2.2 Literature Review

Optical networks are employed to facilitate reliable and faster communications for data transfer. In the recent years, the improvement of communication systems in distributed computing and storage-systems has received some attention. A virtualized optical network is a promising candidate for reliable and cost effective cloud computing environment. The studys about protection and dimensioning of optical networks applying unicast approach is described in Section 2.2.1, and some work considered protection on logical/virtual topologies is listed in Section 2.2.2. In Section 2.2.3, we go through papers that focus on network survivability in cloud/grid environment.

2.2.1 Protection and Rerouting with Unicast

In the past years, several heuristics and algorithms have been reported for protection and rerouting in networks.

Schupke and Prinz [49] approach the design of an optical network in order to minimize the average loss caused by dual failures of fiber links, while single failures are still fully survived. High dual failure restorability is their primary aim, capacity is optimized in a second step. For WDM networks with full wavelength conversion, they formulated a mixed integer linear programming model for dedicated path protection, shared (backup) path protection, and path rerouting with and without stub-release. For larger problem instances in path rerouting,they proposed two heuristics.

They implemented their models with several random networks with 20-26 edges, and their computational results indicate that the connectivity is of much more importance for high restorability values than the overall protection capacity. Shared protection has similar restorability levels as dedicated protection while the capacity is comparable to rerouting. Rerouting surpasses the protection mechanisms in restorability and comes close to 100% dual failure survivability. Compared to single failure planning, both shared path protection and rerouting need significantly more capacity in dual failure planning.

Zhong *et al.* [60] proposed failure inferencing based fast rerouting (FIFR) approach that exploits the existence of a forwarding table per linecard, for lookup efficiency in current routers, to provide fast rerouting similar to Multiprotocol Label Switching (MPLS), while adhering to the destination-based forwarding paradigm.

In their previous work, they have shown that the old approach can only deal with single link failures. In this paper, they extend the FIFR approach to ensure loop-free packet delivery in case of single router failures. Also, thus mitigating the impact of many scenarios of failures. They demonstrated that the proposed approach not only provides high service availability but also incurs minimal routing overhead. They have proved that when a node fails, FLFR, guarantees loop-free forwarding of a packet to its destination if there exists a path to it without the failed node. They have also shown that by inferring node failures, FLFR can handle link failures also without any perceptible increase in the path length stretch.

As for network rerouting against failures, in [43], Nelakuditi *et al.* proposed a local rerouting based approach called failure insensitive routing. The proposed approach prepares for failures using interface-specific forwarding, and upon a failure, suppresses the link state advertisement and instead triggers local rerouting using a backwarding table. With this approach, when no more than one link failure notification is suppressed, a packet is guaranteed to be forwarded along a loop-free path to its destination if such a path exists in the network.

In their paper, they have not only presented a proactive failure insensitive routing approach as an alternative to the reactive approach of the existing link state routing protocols such as OSPF (Open Shortest Path First)/ISIS (Intermediate System-to-Intermediate System) for failure resiliency. Also they described how FIR (failure insensitive routing) approach prepares for failures by computing interface-specific forwarding and backwarding tables, and proved that it ensures reachability of packets to their destinations through local rerouting while suppressing transient single link failures. They have developed a formal model to analyze the routing stability and network availability under both proactive and reactive approaches,and validated it through simulations. Moreover, they have shown that FIR provides better stability and availability than OSPF across various failure frequencies, convergence delays, and network sizes. They experimented their model with topologies of different size. The number of nodes in these topologies is up to 200, and the average node degrees varies from 4 to 6. The results indicate that the improvement due to FIR is markedly better when link failures are frequent and transient. There are several issues related to FIR that require further investigation. The schemes presented assume a forwarding table per each interface and are applicable to single area networks of point-to-point links with symmetric weights.

These papers are giving algorithms or heuristics consider unicast as their routing method which is not suitable in cloud context. Next, we will go through some papers studied the failure protecting problem on virtual topologies.

2.2.2 Resilient Virtual Topologies Mapping in Optical Networks

A survivable virtual topology problem does not have really stringent requirements, and it could be simply described as a survivable mapping problem that how to guarantee the connectivity of a topology while a failure occurs.

The concept "survivable "was first introduced by Modiano *et al.* [41] to describe a routing that can guarantee a network will not be disconnected by a single link failure. In this paper, a condition similar to max-flow min-cut theorem for survivable routing is given as well as an ILP model based on the condition. They compared their ILP model with a shortest path solution on a 14-node, 21-link NSFNET physical topology, and proved that their ILP model can provide better protection against single failure. The same results were got from experiments on bi-directional ring logical topologies with only 6 and 10 nodes, and they leave the search for an efficient solution to the ILP problem as future work.

He *et al.* [28] propose a sub-reconfiguration technique in order to rearrange the paths for WDM (Wavelength Division Multiplexing) networks, using pre-computed alternate backup paths. They report a 10% bandwidth saving with simulation experiments using OPNET. Their algorithm is not scalable enough, and they only did experiments with up to a hundred traffic load.

Todimala *et al.* [55] studied the survivable virtual topology routing under single node/Shared Risk Link Group (SRLG) failure model and present an improved ILP formulation which is originally developed by Modiano *et al.* [42]. However, their formulation is not scalable as well. As a consequence, a general graph (even a medium-sized one) is too large to be solved.

In these papers, the logical topology is given and could not be changed. A study that enable a survivable mapping or reduce the minimal survivable mapping cost by adding logical links into a original topology is presented by Liu *et al.* [35]. Unlike the original survivable mapping problem, they allow the given logical topology to be augmented by adding new logical links to it, and give two reasons to explain the significance of their new survivable mapping problem. First, if the given logical topology does not have a survivable mapping, logical links can be added to it to enable a survivable mapping. Second, if the given logical topology has a survivable mapping, it is possible to reduce the minimal survivable mapping cost by adding logical links to the given logical topology.

They provided a ILP formulation (ILP1) to solve the survivable mapping problem, and then a second ILP model (ILP2) to find a solution to this problem by only adding reflective logical links to achieve the minimal cost. They used a 14-node 21-link NSFNET and a 12-node 18-link random graph in their simulations, and compare their second model (ILP2)

with the ILP model provided in [41]. The results shown that ILP2 obtain an insignificant improvement, and works better on sparser logical topologies than denser ones. The issue of the study is that due to the presence of the exponential number of constraints, their proposed ILP model is not scalable too.

Most of the proposed models or algorithms do not have ability to deal efficiently with general cases. A model that could handle larger instances was designed by Jaumard *et al.* [30]. They proposed a new optimization model, an enhanced cutset model which can against either single or multiple failures in IP-over-WDM. One additional model based on a multi-flow formulation is presented in [31]. Both models can solve exactly most benchmark instances, which were only solved heuristically so far. But these two are mainly focus on logical topologies' survivability, and doesn't pay attention to resourc minimization. Bandwidth requirements is considered in [29], which is also the objective of our current RVNM problem. In this paper, Jaumard *et al.* provide a full recovery for IP requests, assuming that all the traffic of a disrupted IP request is sent over a single restoration path. They investigate in detail the respective bandwidth requirements of the two extreme cases: under the assumption of single or multiple link/node failures. They experimented their models by using four different topologies, and the size of these topologies is up to 45 edges and 24 nodes. The traffic they generated on these topologies is up to 40 units, which is not large enough. Time-varying traffic was not been considered in these studies.

2.2.3 Protection in Cloud/Grid Environment

The problem of dimensioning optical clouds/grids basically involves finding the amount of resources (network and servers), to meet a set of given cloud services (i.e., traffic requests). The main complication herein stems from the anycast principle: in a cloud scenario, we have a certain flexibility in choosing an appropriate data center among a given set of possible locations to serve the cloud traffic. Thus, the classical notion of a (source, destination)-based traffic matrix disappears [20] [9].

Christodoulopoulos *et al.* [14] presented an initial study which proposes models that reflect real-world optical grid application traffic characteristics, appropriate for simulation purposes. They addressed scheduling and routing algorithms in dimensioning problems of optical grids.

They concluded that: *(i)* Job inter-arrival times on the observed Grid level can be successfully modeled by a Poisson process, but on the Grid site level the long range dependency needs to be taken into account and Homogeneous Poisson (HP), Markov Modulated Poisson Process (MMPP) or Pareto-Exponential (PE) models need to be used. *(ii)* For the job execution times, they achieved the most satisfactory results with a hyper-exponential process.

As for the protection of optical grids, Zang *et al.* [59] proposed path protection routing and wavelength assignment ILP formulation for optical grid networks. The objective is to minimize the number of bandwidth units for a given (fixed) destination for the working path and anycast principle is used for the backup path. This ILP formulation successfully solves only instances with up to size of 20 requested connections on Geant2 network topology (17 nodes, 45 directional links), due to scalability problems. The path protection under single-link-failure survivability has been considered. An experimental result shows that 20% of the number of wavelengths have been saved as compared to the case, where destination is given (fixed) in both working and backup paths [50].

Shaikh *et al.* [51] presented two ILP models focus on resiliency against single link failure in optical grids. The problem is solved by providing primary and backup baths for a set of requests, and guarantee that these requests can always be able to reach an operational data center through either a primary path or backup path. The first model they provided lack of scalability, and the second one which involve column generation works well with large-scale optimization. Three 28-node mesh networks with 35, 41 and 59 links respectively are used in their experiments, and the number of requests is up to 400. They found that for lower node degrees, the potential bandwidth saving are much higher, and these savings of their relocation strategy come at the price of increased load on the relocation servers.

Develder *et al.* [17] addressed the dimensioning problem of optical grids, decided how much server infrastructure need to provide, at which locations in a given topology need to deploy. Network and server capacity is also taken into consideration. In this paper, they presented an elegant and scalable model to jointly dimension network and server capacity for grid-like scenarios, where demands for IT infrastructure (servers) and connectivity towards it arise with a freedom in choosing the IT resource location (anycast principle). This model is using column generation approach, allows providing resilience against both network and

server site failures by specifying the appropriate shared risk link groups (SRLGs) comprising possibly jointly failing resources. They evaluated the approach in a case study on a 28-node 41-link European network. There, relocation in protection against single link failures achieves network resource savings around 19%, but calls for around 11% extra server capacity. Providing also protection against server site failures incurs 55% extra servers , and 26% extra wavelengths.

In their follow-up study [16], they provided a new integer linear programming (ILP) formulation to solve the resilient grid/cloud dimensioning problem using failure-dependent backup routes. They considered their previous work as using failure-independent path rerouting approach, thus, for a given demand unit the alternate path (the backup path) under any failure condition affecting the primary path was always the same. They use same test network as previous, and compared four different cases for failure-dependent backup rerouting. The number of requests in their experiments up to 350. However, the results shown that in the anycast routing problem, the benefit of using failure-dependent (FD) rerouting is limited compared to failure-independent (FID) backup routing.

However, these previous works did not consider any resource to accommodate synchronization between distinct working and backup data center locations (as opposed to this thesis).

In [58], Vizcano *et al.* proposed a algorithm to protect optical transport networks with fixed and flexible grid. This article evaluates the energy and cost efficiency of an innovative flexible grid orthogonal frequency division multiplexing (OFDM) based network and compares them with those for conventional wavelength division multiplexing (WDM) networks. Due to the importance of resilience in optical transport networks, their study considers and evaluates different protection schemes. The results demonstrate the potential energy efficiency improvements that can be achieved by an elastic OFDM-based technology, especially when a shared protection scheme is adopted, and give an insight into the potential cost benefits that such a novel technology can offer to telecommunication carriers.

One of the studies about resilience in cloud context is provided by Bui *et al.* [11]. They investigated the design of scalable optimization models to perform the virtual network mapping resiliently for both link and node failures. In this paper, they focused on the planning

of the core network as well as allocation of server capacity at data centers. Both optical network and data centers are assumed to be virtualized. Two models basic on different resilience options were presented, one used VNO-resilience and another one used PIP-resilience. They evaluated their models on a 24-node US nationwide backbone network with 43 non-directional links and the number of requests in their test data set is varying from 10 to 40 which is really small. From the results, thdy didn't find intuitively difference between VNO-resilience and PIP-resilience considering bandwidth demand.

Most of studies only considered the primary (working) path and back path when design routing to against network failures. When a failure happens, switch to backup path stiffly may cause some quality issues for customs which is hard to accepted today. In our models, we considered the synchronization between primary and backup path. Therefore, after switch the delivery could be continued exactly from where it was interrupted. In addition, as our best knowledge, we are the first one who work on time-varying anycast traffic exploiting protection rerouting or reconfiguration. Yet, there we assumed an iterative approach: we formulated the optimization problem to find the best routing going from one period to the next.

Chapter 3

Statement of the Project

In this chapter, we provide the description of our project and explain the basic idea about our mapping scheme. In Section 3.1, we introduce our VNO-resilience model, analyze different failure situations, explain the time-varying traffic we considered in this thesis and our back-up sharing strategy. The definition of the variables and parameters used in our model is described in Section 3.2.

3.1 Outline

In this section, we will introduce our VNO-resilience model, analyze different cases of failures and give a detailed description of our Resilient Virtual Networking Mapping (RVNM) project.

3.1.1 Virtualization and Resilience in Cloud Computing

In this thesis, we focus on resilient virtual topology mapping: how to decide on what routes to follow in the physical network to map the virtual connections from source nodes to data centers and which part of the applications are being served? The cloud services' requests are offered by a virtual network operator (VNO), which runs its virtual network (VNet) on top of the physical network resources that offered by a physical infrastructure provider (PIP). The problem we addressed is how to determine a resilient VNet topology that minimizes the bandwidth resources which are requested by the VNO to the PIP in time-varying traffic. We assume a VNO-resilience scheme, i.e., rerouting in the virtual network under the VNO control

(see Section 3.1.2, or, e.g., [10]). We design the VNet such that requests can survive through single failures, which can affect either the physical network or data center infrastructure.

3.1.2 VNO-resilience

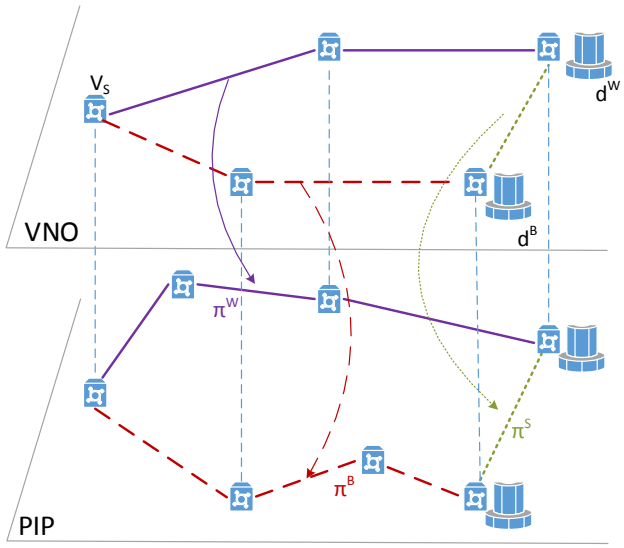

Fig. 3.1: The VNO-resilience Scheme.

The VNO-resilience model we adopt is illustrated in Fig. 3.1: it provides 1:1 protection routing in the VNet for network failures, where the working and protection paths of a service have to be physically link/node disjoint (if only the protection against link failures is requested, then only link disjoint condition needs to be satisfied. Same as node disjoint, if only the node protection needs to be ensured). The working path (π^{W}) routes the services from their source node (v_s) towards the primary DC (d^{W}), the protection path (π^{B}) towards the backup DC (d^{B}), while π^{W} and π^{B} are disjoint in their physical layer mapping. In addition, a synchronization path (π^{S}) is established in order to handle migration and failure routing requirements when a DC failure occurs: services then need to be rerouted from the primary d^{W} to backup d^{B}. Thus, the resulting VNet for the request from source v_s comprises three virtual paths, mapped to the physical $\pi^{\mathrm{W}}, \pi^{\mathrm{B}}$ and π^{S} paths, respectively. Note that both π^{W} and π^{B} need to carry the overall traffic (only when π^{W} or d^{W} are affected by a failure, then

π^{B} need to carry the traffic), but π^{S} possibly only a fraction thereof, since it is used just to keep the state at the backup location d^{B} which synchronized with d^{W} (or vice versa), in order to allow smooth migration upon d^{W} failure (or recovery).

Further, we assume that there is an automatic switch-back to the original network path and DC once a fault is repaired, and therefore we allow reusing the same network/DC capacity to protect against other failures: backup capacity is shared. Under the assumptions that (A1) the backup DC has a different location for the primary DC, (A2) π^{W} and π^{B} are link disjoint, (A3) π^{W} and π^{S} are link disjoint, protection is guaranteed against any single link failure and any single DC failure. We now qualitatively discuss the various failure cases that we protect against:

(i) **Failure of link** $\ell \in \pi^{\mathrm{W}}$: the request is rerouted to the backup data center d^{B}, using the backup path π^{B} (which is link disjoint from π^{W}, thus $\ell \notin \pi^{\mathrm{B}}$). If $\ell \in \pi^{\mathrm{S}} \cap \pi^{\mathrm{W}}$, then as long as the failure is not restored, the primary data center d^{W} cannot be kept in sync with the now operational d^{B}. Thus, right after the reparation of ℓ, the primary d^{W} is in stale state, and hence it will switch back to d^{W} because either it suffers from this stale state or it need to wait some extra time to handle the requests again. The remedy is to enforce $\pi^{\mathrm{W}} \cap \pi^{\mathrm{S}} = \emptyset$. (Yet, note that the same issue of a non-synchronized primary d^{W} also occurs after the reparation of d^{W} which itself failed.)

(ii) **Failure of link** $\ell \in \pi^{\mathrm{S}} \setminus \pi^{\mathrm{W}}$: there is no immediate issue. Yet, if shortly after ℓ is repaired and working path π^{W} fails, the switchover to the backup d^{B} (via path π^{B}) suffers from stale state since the failing π^{S} interrupt the synchronization between primary and backup DCs. This can only be remedied by providing a second synchronization path π^{S}, that is link disjoint with π^{S}.

(iii) **Failure of link** $\ell \in \pi^{\mathrm{B}}$: again no immediate problem arises (since this means that π^{W} is operational, given $\pi^{\mathrm{W}} \cap \pi^{\mathrm{B}} = \emptyset$). However, if $\ell \in \pi^{\mathrm{S}} \cap \pi^{\mathrm{B}}$ and shortly after ℓ repair the primary path π^{W} (or d^{W}) failures — meaning that now π^{B} is followed towards d^{B} — the secondary data center d^{B} might not be fully synchronized yet. Clearly, this can be remedied by choosing $\pi^{\mathrm{B}} \cap \pi^{\mathrm{S}} = \emptyset$. Yet, the issue is similar to the one of case (ii), which obviously remains, even if we take $\pi^{\mathrm{S}} \cap \pi^{\mathrm{B}} = \emptyset$.

(iv) **Failure of primary DC** d^{W}: requests are rerouted to backup d^{B} via the π^{B} path. Clearly, the failing of d^{W} can not be kept in sync with the operational backup d^{B}. Thus, we might need to wait some time after d^{W}'s repaired to switch back requests via π^{W}. Any failure that would occur shortly after d^{W}'s reparation and which would prevent services to remain being served at d^{B} could imply service degradation because of the unsynchronized d^{W}: (a) failure of π^{S}, (b) failure of π^{B}, or (c) failure of d^{B}. However, protection against such a failure event requires extra DC resources or extra paths.

3.1.3 Time-varying Traffic

This project investigates the problem that whether it is worth reconfiguring the primary and the backup paths in order to save bandwidth when the communication traffic patterns change. Note that this change is not necessarily limited to a scaling of the volume, but also its geographical distribution: when considering large backbone networks (as the ones that we are designing VNets over), they might comprise different time zones where activities are shifted in time, and hence the resulting volume of cloud requests fluctuates differently.

As changing the VNet mapping operations may have an impact on the real-time performance of the cloud requests they are servicing, we propose to investigate three scenarios:

- In *Scenario I* (very conservative), we do not allow reconfiguring in the already established paths. In this scenario, once a configuration is assigned to the traffic, these traffic is not allowed to use other configurations anymore. However, the troublesome rerouting work can be avoided, we will get higher bandwidth cost;

- In *Scenario II* we only allow reconfiguring backup and/or synchronization routes (π^{B} and/or π^{S}) for traffic that continues from one period to the next. By allowing back-up rerouting, we may gain more bandwidth sharing. But the bandwidth cost probably is still not the minimum one, since the primary (working) paths are not allowed to be rerouting;

- In *Scenario III* we assume complete freedom and thus also allow to change the primary paths (π^{W}). Yet, we always look for the optimal solution (in terms of minimal link bandwidth consumed on the PIP layer) with the lowest number of configuration changes. The freedom of both working and back-up paths rerouting may lead us to the best bandwidth

cost, but on the other hand, change working paths can cause the interruption of data delivery.

3.1.4 Bandwidth Sharing

The number of Internet users in the world is already more than 3,100 million, and keep increasing every second. In one second, there were 48,318 Google searches, 98,898 YouTube videos viewed, 2,385,699 emails sent [3]. As traffic through network going to be crowded, a good mapping scheme is essential. Under the failure-protecting condition, one of the efficient way to achieve bandwidth saving is backup sharing.

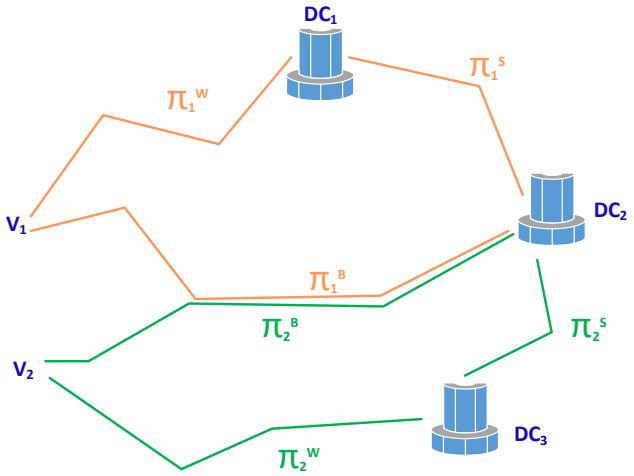

Fig. 3.2: Backup Sharing Example.

An example is showed in Fig. 3.2. Suppose we have a service request k_1 asked by a user located in v_1, which uses π_1^{W} as primary path, π_1^{B} as backup path, and π_1^{S} as synchronization path. Assuming that the bandwidth need for k_1 is β_1, and the synchronization path need β_1^{S}. Another service request k_2 is located at v_2 with a need of β_2 bandwidth and β_2^{S} for synchronization. π_2^{W}, π_2^{B}, π_2^{S} are used by k_2 as primary, backup and synchronization paths respectively. Therefore, the bandwidth need of this topology is the sum of the primary part $\beta_1 + \beta_2$, the synchronization part $\beta_1^{\mathrm{S}} + \beta_2^{\mathrm{S}}$, and the backup part $\max\{\beta_1, \beta_2\}$, instead of $2(\beta_1 + \beta_2) + \beta_1^{\mathrm{S}} + \beta_2^{\mathrm{S}}$ without backup sharing.

3.2 Notations and Statements

In this section, we define all the notations for the four mathematical models which will be presented in Chapter 4. The first two are non aggregated traffic models with link formulation and path formulation respectively, and the last two are aggregated traffic models. There are some difference between notations used for the these four models. For the parameters and variables which are used differently, we will give some specific explanations.

3.2.1 Network, Time Periods and Traffic

The cloud network is modeled by an undirected graph $G = (V, L)$ where V is the node set (indexed by v) and L is the link set (indexed by ℓ), for which $\omega(v)$ denotes the set of links adjacent to v, and $\|\ell\|$ is the length of link ℓ.

In this network, there is a set of data centers V_{D} (indexed by d). The current model assumes (at most) a single data center per node.

In order to solve the RVNM problem for multiple time periods, let T be the set of time periods. And T' is the set of time periods without the first period, which can be defined as following:

$$T' = T \setminus \{t_1\}$$

, where t_1 represents the first time period.

Traffic is defined by the number of service requests (demands), originating from a set of source/service nodes V, with generic index v.

The set of service requests is represented by K, and K can be defined as:

$$K = \bigcup_{v \in V} K_v \qquad \text{OR} \qquad K = \bigcup_{v \in V} \bigcup_{t \in T} K_{v,t}$$

where K_v is the set of requests associated with source node $v \in V$, and $K_{v,t}$ is the set of requests associated with source node $v \in V$ which are "alive" at time period t.

Each request k is characterized by its bandwidth requirement Δ_k, its source (or origin)

v_k, its "alive" time periods, and δ_k (with $0 \leq \delta_k \leq 1$) representing the fraction of Δ_k that is required for synchronization between the primary and the backup data center.

In last two models, we aggregated requests with same source node, so in these two models parameters Δ_k and δ_k will be replaced by $\Delta_{v,t}$ and δ_v. $\Delta_{v,t}$ represents bandwidth requirement of all the "alive" requests originating from node v at time period t, and $\Delta_{v,t}$ represents the fraction of bandwidth requirement that is required for synchronization for all the requests with source node v.

T'_k represents the set of time period associated with request k, which can be defined as follows:

$$T'_k = \left(t_k^{\text{START}}, t_k^{\text{END}}\right]$$

where t_k^{START} is the time period where request k appears, and t_k^{END} is the time period where k ends.

3.2.2 Paths

Let Π be the set of paths which include all the working (primary) paths, backup paths and synchronization paths. Each path $\pi \in \Pi$ is characterized by its source node v_k and a set of links associated with it. Also, Π can be represented as :

$$\Pi = \bigcup_{v \in V_{\text{s}}} \Pi_v$$

where Π_v is set of paths associated with source node $v \in V_{\text{s}}$.

3.2.3 Configurations

The mathematical model we proposed relies on the notion of configurations, where a configuration is associated with a set of service requests originating at a given source node. Let C be the overall set of configurations:

$$C = \bigcup_{v \in V_{\text{s}}} C_v,$$

where C_v is the set of configurations associated with source node $v \in V_{\text{s}}$.

We define a configuration $c \in C_v$ by:

(i) a set of 3 paths, one primary path π^{W} originating at v_s towards a primary data center d^{W}, one backup path π^{B} originating at v_s towards a primary data center d^{B}, and one synchronization paths (π^{S}) between the primary and the backup data center.

(ii) the service requests are routed and protected by this set of 3 routes.

There are another two notations being used in the models with path formulations: $C^{\mathrm{W},\pi}$ and $C^{\mathrm{BS},\pi,\pi'}$. $C^{\mathrm{W},\pi}$ is defined as the set of configurations which use path $\pi \in \Pi$ as working path. $C^{\mathrm{BS},\pi,\pi'}$ is the set of configurations which use π as backup path and π' as synchronization path.

3.2.4 Parameters and Variables for Master Problems

Recall CG problem, each problem is decomposed into master problem and pricing problem. Here, since we use CG to solve our RVNM problem, we have two sub-models for each mathematic model: one for master problem and one for pricing problem.

Master problems are aimed to choose configurations for each service request, and the parameters used in models for master problems are as follows:

- $p_\ell^{\mathrm{W},c} = 1$ if link ℓ is used by the working path of configuration c, 0 otherwise;
- $p_\ell^{\mathrm{B},c} = 1$ if link ℓ is used by the backup path of configuration c, 0 otherwise;
- $p_\ell^{\mathrm{S},c} = 1$ if link ℓ is used by the synchronization path of c between the primary data center and the backup data center, 0 otherwise;
- $a_v^{\mathrm{W},c} = 1$ if node $v \in V_{\mathrm{D}}$ is selected as the primary data center by configuration c, 0 otherwise;
- $\alpha^{\mathrm{W},c,\pi} = 1$ if path π is used by configuration c as a working path, 0 otherwise;
- $\alpha^{\mathrm{B},c,\pi} = 1$ if path π is used by configuration c as a backup path, 0 otherwise;
- $\alpha^{\mathrm{S},c,\pi} = 1$ if path π is used by configuration c as a synchronization path, 0 otherwise;

The variables need to be used by master problems are:

- $z^c_{k,t} = 1$ if configuration c is selected by request k at time period t, 0 otherwise;
- $x^{\mathrm{W}}_{k,t} = 1$ if request k select different working paths at time period t and $t-1$, 0 otherwise;
- $x^{\mathrm{BS}}_{k,t} = 1$ if request k select different backup paths or synchronization path at time period t and $t-1$, 0 otherwise;
- BW is total bandwidth needed on the network in order to satisfy every service request.
- $\beta^{\mathrm{W}}_{\ell,t} \in \mathbb{R}$: the working bandwidth on ℓ;
- $\beta^{\mathrm{B}}_{\ell,t} \in \mathbb{R}$: the backup bandwidth on ℓ;
- $\beta^{\mathrm{S}}_{\ell,t} \in \mathbb{R}$: the bandwidth of Synchronization path on ℓ;

In last two models, we aggregated the requests with same source node. Therefore, there are several variables which are different from the first two models listed as follows:

- $z^c_t \in \mathbb{R}$: the bandwidth assigned to configuration c at time period t;
- $x^{\mathrm{W}}_{v,t} \in \mathbb{R}$: the bandwidth of changed working paths originating on source node $v \in V_{\mathrm{s}}$ at time period t;
- $x^{\mathrm{BS}}_{v,t} \in \mathbb{R}$: the bandwidth of changed backup paths or synchronization paths associated requests originating on source node $v \in V_{\mathrm{s}}$ at time period t.
- $\Delta_{v,t} \in \mathbb{R}$: the amount of traffic that originating on source node $v \in V_{\mathrm{s}}$ at time period t.
- $\Delta^{\mathrm{DEL}}_{v,t,t-1} \in \mathbb{R}$: the amount of traffic that exist at time period $t-1$, and is deleted at time period t which is originating on source node $v \in V_{\mathrm{s}}$.
- $\Delta^{\mathrm{ADD}}_{v,t,t-1} \in \mathbb{R}$: the amount of traffic that does not exist at time period $t-1$, and shows up at time period t which is originating on source node $v \in V_{\mathrm{s}}$.

3.2.5 Variables for Pricing Problems

Pricing problems in this project is to generation configurations, the variables used in models for pricing problems are as follows:

- $p_\ell^{\mathrm{W}} = 1$ if link ℓ is used by the working path of the configuration under construction, 0 otherwise;
- $p_\ell^{\mathrm{B}} = 1$ if link ℓ is used by the backup path of the configuration under construction, 0 otherwise;
- $p_\ell^{\mathrm{S}} = 1$ if link ℓ is used by the synchronization path of the configuration under construction between the primary data center and the backup data center, 0 otherwise;
- $a_v^{\mathrm{W}} = 1$ if node v is selected as a data center location by the working path in the configuration under construction, 0 otherwise;
- $a_v^{\mathrm{B}} = 1$ if node v is selected as a data center location by the backup path in the configuration under construction, 0 otherwise;
- $d_v^{\mathrm{W}} = 1$ if node v is on the working path in the configuration under construction, 0 otherwise;
- $d_v^{\mathrm{B}} = 1$ if node v is on the backup path in the configuration under construction, 0 otherwise;
- $d_v^{\mathrm{S}} = 1$ if node v is on the synchronization path between the primary data center and the backup data center in the configuration under construction, 0 otherwise;
- $\alpha^{\mathrm{W},\pi} = 1$ if path π is the working path in the configuration under construction, 0 otherwise;
- $\alpha^{\mathrm{B},\pi} = 1$ if path π is the backup path in the configuration under construction, 0 otherwise;
- $\alpha^{\mathrm{S},\pi} = 1$ if path π is the synchronization path in the configuration under construction, 0 otherwise.

In this chapter, we proposed a detailed description of our RVNM project. Our main goal is to solve a mapping problem in order to protect against single network failure as well as to optimize bandwidth requirement. We also investigate three scenarios to figure out whether it is worth to reconfigure the primary and backup paths. The parameters and variables that listed in this chapter are used in our mathematical models which will be presented next.

Chapter 4

Models for the Problem

In this chapter, we propose four models for our RVNM problem. Section 4.1 gives a general introduction to these models. The first two models are non-aggregated models with link formulations and path formulations respectively. Here, the term "non-aggregated" means the traffic is defined by the number of service requests and is not gathered by their source node, so that configuration assignment is based on each request. The details of these two models is in Section 4.2 and Section 4.3. After that, there will be two aggregated models. In these two models, we gather all the requests together in the same source node and select multiple configurations for them. The detail description is presented in Section 4.4 and Section 4.5, respectively. We compare these four models and summarize the advantages and disadvantages for each model in Section 4.6.

4.1 Introduction

Our models are based on column generation technique. As explained previously, our RVNM problem is separated into a master problem and a pricing problem. The master problem is aimed to select configurations for all the service requests in order to minimize bandwidth requirement on the network. And the pricing problem is to generate configurations which are provided to master problem and help it to get better results.

As shown in Fig. 4.1, each time RMP solve the problem with existing set of configurations, it gives the values of the dual variables to PP. PP uses these dual values to generate a

configuration with 3 paths which may help RMP to improve the previous solution (in our case, to get smaller bandwidth requirement). After adding this configuration into the configuration set, RMP will solve the problem again. This loop will keep going until PP cannot find any configuration to improve the result of RMP, i.e. the result of PP is positive. Then the ILP will be solved based on RMP solution. In our aggregated models, it's not necessary to do the ILP step, since none of the decision variables need to be integer.

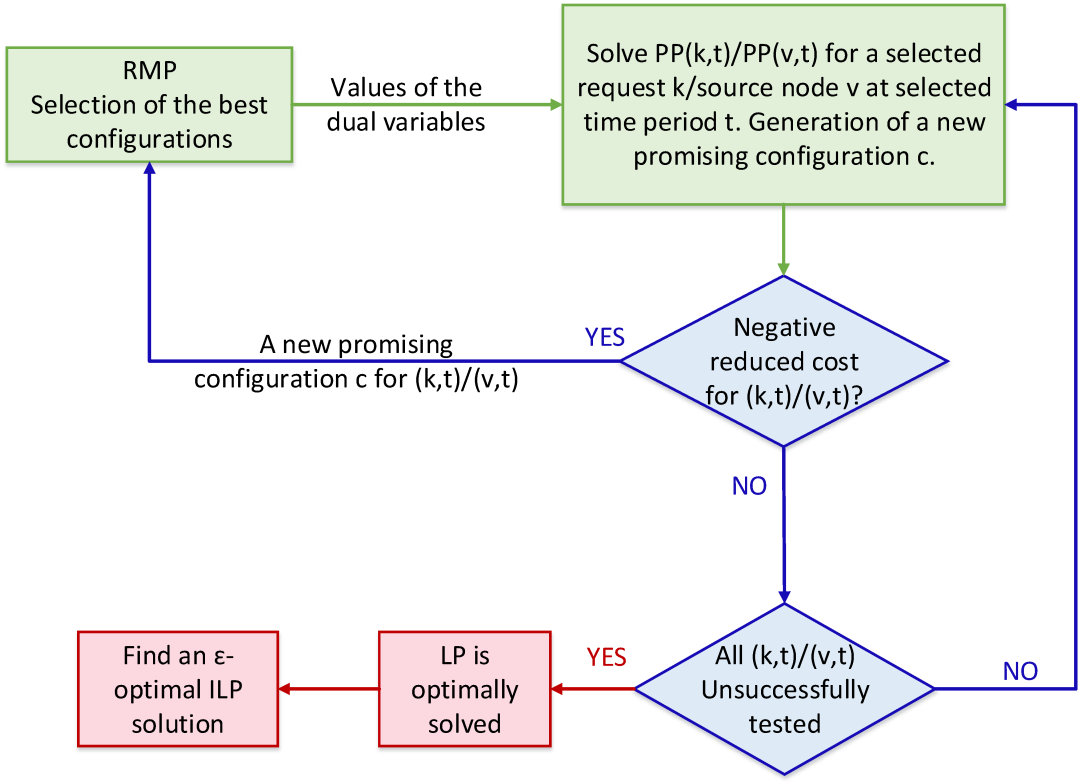

Fig. 4.1: General Column Generation Flowchart for RVNM Problem.

4.2 Model 1: Non Aggregated Traffic & Link Formulation

The purpose of this model is to find out a mapping solution for a given network and set of requests, and this solution should have a minimized bandwidth requirement as well as the number of reconfigurations. Let's take an example to explain it clearly. Assuming a network as shown in Fig. 4.2 is given, and requests k_1 and k_2 are two requests in the given requests set and we are considering *Scenario II* (in which only backup paths have freedom to change, see Section 3.1.3). In order to achieve the optimal bandwidth requirement, at time period t_1, configuration C_1 which contains working (primary) path C_1^{W}, backup path C_1^{B}, synchronization path C_1^{S}, working (primary) data center DC_1 and backup data center DC_2 is assigned to request k_1, and configuration C_2 is assigned to k_2. At time period t_2, since there are some new requests appear, we could have two options to get the optimal bandwidth requirement. Option 1 need to reconfigure both k_1 and k_2, whereas only k_1 need to be reconfigure in Option 2. Therefore, in order to achieve our purpose, Option 2 will be selected by this model. The parameters and variables for this model is listed previously in Section 3.2.4.

4.2.1 Master Problem

The master problem is to choose configurations for the given requests from a configuration set. In this model, each request will choose one configuration (if there is one has same source node in the configurations set). The minimization of configuration changes is considered for *Scenario II* and *Scenario III*, but have less priority than the optimization of bandwidth requirement. These changes are detected by checking every link in the network.

4.2.1.1 Objective

$$\min \mathrm{BW} \quad + \quad \mathrm{PENAL}^{\mathrm{DISRUPT\_B}} \sum_{k \in K} \sum_{t \in T'_k} x^{\mathrm{BS}}_{k,t} \quad + \quad \mathrm{PENAL}^{\mathrm{DISRUPT\_W}} \sum_{k \in K} \sum_{t \in T'_k} x^{\mathrm{W}}_{k,t}. \quad (4.1)$$

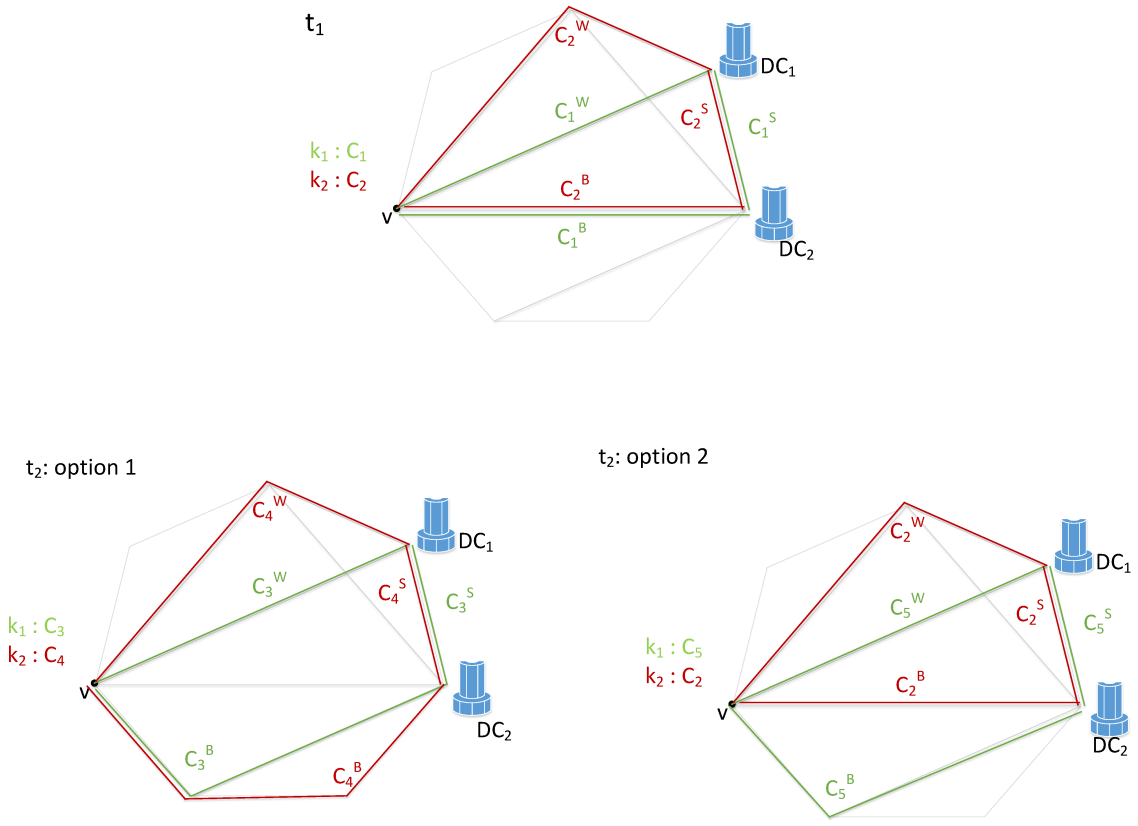

Fig. 4.2: Mapping Example for Model 1

where BW is the maximum bandwidth requirements among all the time period, which can be defined as constraint (4.2).

$$\mathrm{BW} = \max_{t \in T} \sum_{\ell \in L} (\beta^{\mathrm{W}}_{\ell,t} + \beta^{\mathrm{B}}_{\ell,t} + \beta^{\mathrm{S}}_{\ell,t}) \cdot \|\ell\| \tag{4.2}$$

The parameters $\textsc{penal}^{\textsc{disrupt\_b}}$ and $\textsc{penal}^{\textsc{disrupt\_w}}$ are to penalize the rerouting of working path and backup path. We need to first minimize the bandwidth requirement, if the optimal solution of bandwidth requirement is achieved, then we will consider to minimize reconfiguration of working path and backup path.

4.2.1.2 Constraints

$$\sum_{c \in C_v} z_{k,t}^{c} \geq 1 \qquad v \in V, k \in K_v, t \in T \tag{4.3}$$

$$\sum_{v \in V} \sum_{k \in K_{v,t}} \sum_{c \in C_v} \Delta_k \, p_{\ell}^{\mathrm{W},c} z_{k,t}^{c} = \beta_{\ell,t}^{\mathrm{W}} \qquad \ell \in L, t \in T \tag{4.4}$$

$$\sum_{v \in V} \sum_{k \in K_{v,t}} \sum_{c \in C_v} \Delta_k \delta_k \, p_{\ell}^{\mathrm{S},c} z_{k,t}^{c} \; = \beta_{\ell,t}^{\mathrm{S}} \qquad \ell \in L, t \in T \tag{4.5}$$

$$\sum_{v \in V} \sum_{k \in K_{v,t}} \sum_{c \in C_v} \Delta_k \, p_{\ell'}^{\mathrm{W},c} \, p_{\ell}^{\mathrm{B},c} \, z_{k,t}^{c} \leq \beta_{\ell,t}^{\mathrm{B}} \qquad \ell' \in L, \ell \in L \setminus \{\ell'\}, t \in T \tag{4.6}$$

$$\sum_{v' \in V} \sum_{k \in K_{v',t}} \sum_{c \in C_{v'}} \Delta_k \, a_v^{\mathrm{W},c} \, p_{\ell}^{\mathrm{B},c} \, z_{k,t}^{c} \; \leq \beta_{\ell,t}^{\mathrm{B}} \quad v \in V_{\mathrm{D}}, \ell \in L, t \in T \tag{4.7}$$

$$\left| \sum_{c \in C_v} p_{\ell}^{\mathrm{W},c} z_{k,t}^{c} - \sum_{c \in C_v} p_{\ell}^{\mathrm{W},c} z_{k,t-1}^{c} \right| \leq x_{k,t}^{\mathrm{W}} \qquad v \in V, k \in K_v, \ell \in L, t \in T_k' \tag{4.8}$$

$$\left| \sum_{c \in C_v} p_{\ell}^{\mathrm{B},c} z_{k,t}^{c} - \sum_{c \in C_v} p_{\ell}^{\mathrm{B},c} z_{k,t-1}^{c} \right| \leq x_{k,t}^{\mathrm{BS}} \qquad v \in V, k \in K_v, \ell \in L, t \in T_k' \tag{4.9}$$

$$\left| \sum_{c \in C_v} p_{\ell}^{\mathrm{S},c} z_{k,t}^{c} - \sum_{c \in C_v} p_{\ell}^{\mathrm{S},c} z_{k,t-1}^{c} \right| \leq x_{k,t}^{\mathrm{BS}} \qquad v \in V, k \in K_v, \ell \in L, t \in T_k' \tag{4.10}$$

$$\mathrm{BW} \geq \sum_{\ell \in L} (\beta_{\ell,t}^{\mathrm{W}} + \beta_{\ell,t}^{\mathrm{B}} + \beta_{\ell,t}^{\mathrm{S}}) \cdot \|\ell\| \qquad t \in T \tag{4.11}$$

$$z_{k,t}^{c} \in \{0, 1\} \qquad c \in C, k \in K_t, t \in T \tag{4.12}$$

$$\mathrm{BW}, \beta_{\ell,t}^{\mathrm{W}}, \; \beta_{\ell,t}^{\mathrm{B}}, \; \beta_{\ell,t}^{\mathrm{S}} \in \mathbb{R} \qquad \ell \in L, t \in T \tag{4.13}$$

Constraint (4.3) make sure that every request will select at least one configuration, and combine with object function (4.1), each request will only select one configuration. Constraints (4.4) and (4.5) compute the working and synchronization bandwidth requirements on link ℓ at time period t, respectively. Constraints (4.6) ensure sufficient shared backup bandwidth requirements on link ℓ at time period t subject to a single link failure. If any requests which share a link ℓ on their backup paths is link disjoint on their working paths, the backup bandwidth on the link ℓ could be larger than or equal to the largest bandwidth requirement among these requests. Otherwise, if these requests also share links on their working path, the backup bandwidth on ℓ should be larger than or equal to the summation of these requests' bandwidth requirements. Constraints (4.7) guarantee sufficient backup bandwidth ℓ to handle any data center failure. If any requests which share link ℓ on their

backup paths also share data center v, the bandwidth of ℓ should be larger or equal to the summation of their bandwidth requirements. Otherwise, we only need to ensure that the largest bandwidth requirement among these requests is smaller than the backup bandwidth on ℓ. Constraints (4.11) defines variable BW which is the overall bandwidth requirements. BW should be no smaller than the bandwidth requirements at any time period. The last two set of constraints, i.e., (4.12) and (4.13), define the domain of the variables.

In *Scenario I*, requests can not be reconfigured, so we add following constraints:

$$x^{\mathrm{W}}_{k,t} = 0, x^{\mathrm{W}}_{k,t} = 0 \quad k \in K_v, t \in T'_k \tag{4.14}$$

These constraints make sure that any request at time period t will choose same configuration as previous time period, if this request survive more than one time period.

In *Scenario II*, only backup and synchronization paths can be modified. So, in *Scenario II*:

$$x^{\mathrm{W}}_{k,t} = 0 \qquad k \in K_v, t \in T'_k \tag{4.15}$$

$$x^{\mathrm{W}}_{k,t} \in \{0,1\} \quad k \in K_v, t \in T'_k \tag{4.16}$$

Constraint (4.15) guarantees that the working path will be fixed for all time period. Constraint (4.16) counts the number of backup-changed requests. The synchronization change is also counted as backup change. If one request changes backup path and synchronization path at same time, we only count it once.

In *Scenario III*, all the paths can be modified:

$$x^{\mathrm{W}}_{k,t}, x^{\mathrm{BS}}_{k,t} \in \{0,1\} \quad k \in K_v, t \in T'_k \tag{4.17}$$

Constraints (4.17) let both working path and backup path (include synchronization path) to be free changed and count the number of working-changed and backup-changed requests, respectively.

If we focus on constraints (4.8) to (4.10), we notice that there are absolute formulations which need to be linearized, and $|a-b| < x$ can be linearized by $x \geq a-b$ and $x \geq b-a$. Another issue is that for the decision variable z, there are both z_t and z_{t-1} shown up in the constraints, and when we write pricing model, we need to consider three different situations: 1) $t \in T'_k, t-1 \notin T'_k$, 2) $t \in T'_k, t-1 \in T'_k$, 3) $t \notin T'_k, t-1 \in T'_k$. Therefore, we need to have

three models for pricing problem, and it's a bit complex for implementation. In order to make it simpler (just have one model for pricing problem), we introduce the variables $\gamma^{\mathrm{W}}_{k,l,t}$, $\gamma^{\mathrm{B}}_{k,l,t}$, $\gamma^{\mathrm{S}}_{k,l,t}$. The constraints (4.8) (4.9) (4.10) can be linearized as the following:

$$\sum_{c \in C_v} p^{\mathrm{W},c}_{\ell} z^{c}_{k,t} = \gamma^{\mathrm{W}}_{k,\ell,t} \qquad v \in V, k \in K_v, \ell \in L, t \in T \tag{4.18}$$

$$\gamma^{\mathrm{W}}_{k,\ell,t} - \gamma^{\mathrm{W}}_{k,\ell,t-1} \leq x^{\mathrm{W}}_{k,t} \qquad k \in K, \ell \in L, t \in T'_k \tag{4.19}$$

$$\gamma^{\mathrm{W}}_{k,\ell,t-1} - \gamma^{\mathrm{W}}_{k,\ell,t} \leq x^{\mathrm{W}}_{k,t} \qquad k \in K, \ell \in L, t \in T'_k \tag{4.20}$$

$$\sum_{c \in C_v} p^{\mathrm{B},c}_{\ell} z^{c}_{k,t} = \gamma^{\mathrm{B}}_{k,\ell,t} \qquad v \in V, k \in K_v, \ell \in L, t \in T \tag{4.21}$$

$$\gamma^{\mathrm{B}}_{k,\ell,t} - \gamma^{\mathrm{B}}_{k,\ell,t-1} \leq x^{\mathrm{BS}}_{k,t} \qquad k \in K, \ell \in L, t \in T'_k \tag{4.22}$$

$$\gamma^{\mathrm{B}}_{k,\ell,t-1} - \gamma^{\mathrm{B}}_{k,\ell,t} \leq x^{\mathrm{BS}}_{k,t} \qquad k \in K, \ell \in L, t \in T'_k \tag{4.23}$$

$$\sum_{c \in C_v} p^{\mathrm{S},c}_{\ell} z^{c}_{k,t} = \gamma^{\mathrm{S}}_{k,\ell,t} \qquad v \in V, k \in K_v, \ell \in L, t \in T \tag{4.24}$$

$$\gamma^{\mathrm{S}}_{k,\ell,t} - \gamma^{\mathrm{S}}_{k,\ell,t-1} \leq x^{\mathrm{BS}}_{k,t} \qquad k \in K, \ell \in L, t \in T'_k \tag{4.25}$$

$$\gamma^{\mathrm{S}}_{k,\ell,t-1} - \gamma^{\mathrm{S}}_{k,\ell,t} \leq x^{\mathrm{BS}}_{k,t} \qquad k \in K, \ell \in L, t \in T'_k \tag{4.26}$$

Constraints (4.18) to (4.20) are equivalent to previous constraints (4.8), constraints (4.21) to (4.23) are equivalent to (4.9), and (4.24) to (4.26) are equivalent to previous constraints (4.10).

4.2.2 Pricing Problem

Recall that the pricing problem (PP) will determine augmenting configurations, i.e., routes for W, B and S paths such that their addition to the restricted master problem will entail an improvement of the optimal value of the current restricted master. Each PP is written for a given source node k and for a given time period t. Parameters Δ_k and δ_k retain their definition for a request k as in the RMP.

The objective of $\mathrm{PP}_1(k,t)$ with $k \in K, t \in T_k$, where T_k is the set of time periods that make request k to keep "alive", and to minimize the reduced cost $\overline{\mathrm{COST}}^1(z_{k,t})$ as obtained

from the RMP, which is defined as:

$$\begin{aligned}\overline{\text{COST}}_1(z_{k,t}) = 0 - u_{v_k t}^{(4.3)} - \sum_{\ell \in L} u_{\ell t}^{(4.4)} \Delta_k \, p_\ell^{\text{W}} - \sum_{\ell \in L} u_{\ell t}^{(4.5)} \Delta_k \delta_k \, p_\ell^{\text{S}} \\ + \sum_{\ell \in L} \sum_{\ell' \in L \setminus \{\ell\}} u_{\ell \ell' t}^{(4.6)} \Delta_k \, p_{\ell'}^{\text{W}} p_\ell^{\text{B}} + \sum_{v \in V_{\text{D}}} \sum_{\ell \in L} u_{v \ell t}^{(4.7)} \Delta_k a_v^{\text{W}} p_\ell^{\text{B}} \\ - \sum_{\ell \in L} u_{\ell v_k t}^{(4.18)} p_\ell^{\text{W}} - \sum_{\ell \in L} u_{\ell v_k t}^{(4.21)} p_\ell^{\text{B}} - \sum_{\ell \in L} u_{\ell v_k t}^{(4.24)} p_\ell^{\text{S}} \quad (4.27)\end{aligned}$$

where v_k is the source node of request k, $u_{v_k t}^{(4.3)} \geq 0$, $u_{\ell t}^{(4.4)} \gtrless 0$, $u_{\ell t}^{(4.5)} \gtrless 0$, $u_{\ell \ell'}^{(4.6)} \geq 0$, $u_{v \ell}^{(4.7)} \geq 0$, $u_{\ell v_k t}^{(4.18)} \gtrless 0$, $u_{\ell v_k t}^{(4.21)} \gtrless 0$ and $u_{\ell v_k t}^{(4.24)} \gtrless 0$ are the values of the dual variables associated with constraints (4.3), (4.4), (4.5), (4.6), (4.7), (4.18), (4.21), (4.24) respectively. The first explicit 0 term stems from the RMP objective, which does not contain the configuration variable $z_{k,t}^c$.

Note that there are two quadratic terms which can be easily linearized through the introduction of two sets of binary variables $p_{\ell \ell'}^{\text{WB}}$ and $p_{v \ell}^{\text{WB}}$ and the addition of the following constraints:

For all $\ell' \in L, \ell \in L \setminus \{\ell'\}$, $v \in V_{\text{D}}$

$$p_{\ell \ell'}^{\text{WB}} \leq p_{\ell'}^{\text{W}} \quad ; \qquad p_{\ell \ell'}^{\text{WB}} \leq p_\ell^{\text{B}} \quad (4.28)$$

$$p_{\ell \ell'}^{\text{WB}} \geq p_{\ell'}^{\text{W}} + p_{\ell'}^{\text{W}} - 1 \quad (4.29)$$

$$p_{v \ell}^{\text{WB}} \leq a_v^{\text{W}} \quad ; \qquad p_{v \ell}^{\text{WB}} \leq p_\ell^{\text{B}} \quad (4.30)$$

$$p_{v \ell}^{\text{WB}} \geq a_v^{\text{W}} + p_{\ell'}^{\text{W}} - 1. \quad (4.31)$$

For all three scenarios, the path and data center variables have to obey the following constraints:

$$\sum_{\ell \in \omega(v)} p_\ell^{\mathrm{W}} = \begin{cases} 1 & \text{if } v = v_k \\ 2\, d_v^{\mathrm{W}} - a_v^{\mathrm{W}} & \text{if } v \in V_{\mathrm{D}}, v \neq v_k \\ 2\, d_v^{\mathrm{W}} & \text{otherwise,} \\ & \text{i.e., } v \in V \setminus V_{\mathrm{D}}, v \neq v_k \end{cases} \tag{4.32}$$

$$\sum_{\ell \in \omega(v)} p_\ell^{\mathrm{B}} = \begin{cases} 1 & \text{if } v = v_k \\ 2\, d_v^{\mathrm{B}} - a_v^{\mathrm{B}} & \text{if } v \in V_{\mathrm{D}}, v \neq v_k \\ 2\, d_v^{\mathrm{B}} & \text{otherwise,} \\ & \text{i.e., } v \in V \setminus V_{\mathrm{D}}, v \neq v_k \end{cases} \tag{4.33}$$

$$\sum_{\ell \in \omega(v)} p_\ell^{\mathrm{S}} = \begin{cases} 2\, d_v^{\mathrm{S}} - a_v^{\mathrm{W}} - a_v^{\mathrm{B}} & \text{if } v \in V_{\mathrm{D}} \\ 2\, d_v^{\mathrm{S}} & \text{otherwise,} \\ & \text{i.e., } v \in V \setminus V_{\mathrm{D}} \end{cases} \tag{4.34}$$

$$p_\ell^{\mathrm{W}} + p_\ell^{\mathrm{B}} \leq 1 \qquad \ell \in L \tag{4.35}$$

$$\sum_{v \in V_{\mathrm{D}}} a_v^{\mathrm{W}} = 1\,;\ \sum_{v \in V_{\mathrm{D}}} a_v^{\mathrm{B}} = 1 \tag{4.36}$$

$$a_v^{\mathrm{W}} + a_v^{\mathrm{B}} \leq 1 \qquad v \in V_{\mathrm{D}} \tag{4.37}$$

$$a_v^{\mathrm{W}}, a_v^{\mathrm{B}} \in \{0, 1\} \qquad v \in V_{\mathrm{D}} \tag{4.38}$$

$$d_v^{\mathrm{W}}, d_v^{\mathrm{B}}, d_v^{\mathrm{S}} \in \{0, 1\} \qquad v \in V \tag{4.39}$$

$$p_\ell^{\mathrm{W}}, p_\ell^{\mathrm{B}}, p_\ell^{\mathrm{S}} \in \{0, 1\} \qquad \ell \in L. \tag{4.40}$$

Constraints (4.32)–(4.34) are the conventional flow constraints for undirected working, backup and synchronization paths. Constraints (4.35) force primary path and backup path to be link disjoint. Constraints (4.36) ensure that each configuration has exactly one primary and one back up data center, while constraints (4.37) force them to be different. Constraints (4.36) combine with constraints (4.32)–(4.34) to make sure that the generated configuration has one primary path, one backup path and one synchronization path linked primary and backup paths. Constraints (4.38)–(4.40) define the domains of the variables.

The drawback of this model is that it has a large number of constraints in the master

which makes it not scalable. And we will discuss this drawbacks and compare it with other models in Section 4.6.

4.3 Model 2: Non Aggregated Traffic & Path Formulation

The purpose of this model is same as the first model (non aggregated traffic& link formulation), we will select one configuration for each request and try to carry out with a optimal bandwidth requirement as well as reconfigurations. The mapping scheme can be explained by same example in Section 4.2 (see Fig. 4.2). The difference of this model, it is that instead of checking every link to detect configuration changes, we check every path of generated configurations.

4.3.1 Master Problem

We have same objective function (4.1) as Model 1, which is to minimize bandwidth requirement and configuration changes. For the constraints, we keep constraints (4.3)–(4.7), and constraints (4.11)–(4.13).

Constraints (4.8), (4.9) and (4.10) can then be rewritten:

$$|\sum_{c\in C_v} \alpha^{\mathrm{W},c,\pi} z^c_{k,t} - \sum_{c\in C_v} \alpha^{\mathrm{W},c,\pi} z^c_{k,t-1}| \leq x^{\mathrm{W}}_{k,t} \quad v\in V, k\in K_v, t\in T'_k, \pi\in\Pi_v \tag{4.41}$$

$$|\sum_{c\in C_v} \alpha^{\mathrm{B},c,\pi} z^c_{k,t} - \sum_{c\in C_v} \alpha^{\mathrm{B},c,\pi} z^c_{k,t-1}| \leq x^{\mathrm{BS}}_{k,t} \quad v\in V, k\in K_v, t\in T'_k, \pi\in\Pi_v \tag{4.42}$$

$$|\sum_{c\in C_v} \alpha^{\mathrm{S},c,\pi} z^c_{k,t} - \sum_{c\in C_v} \alpha^{\mathrm{S},c,\pi} z^c_{k,t-1}| \leq x^{\mathrm{BS}}_{k,t} \quad v\in V, k\in K_v, t\in T'_k, \pi\in\Pi_v \tag{4.43}$$

Just as we described in Model 1, in *Scenario I* we have constraints (4.14). In *Scenario II* we have constraints (4.15) and (4.16). And for *Scenario III* we keep constraints (4.17).

Constraints (4.41) force each request k select same working path (or count the working path changes in scenario 3) at time period t and the previous time period by checking every path originating at the same node as request k. As illustrated in Fig. 4.3, since we are

considering *Scenario II*, in t_2 if path C_2^{W} is selected for k_2, this constraints force same path must be selected for k_2 in t_1 as well. If we focus on *Scenario III*, and find out that path C_2^{W} is selected in t_1 but not in t_2, then $x_{k,t}^{\mathrm{W}}$ will be set to 1 which means there is a working path rerouting happens for k_2 in time period t_2. Constraints (4.42) and (4.43) is related to backup paths and synchronization paths, respectively.

These three constraints need to be linearized. Similar as what we did in Model 1, we introduce variables $\gamma_{k,t}^{\mathrm{W},\pi}$, $\gamma_{k,t}^{\mathrm{B},\pi}$ and $\gamma_{k,t}^{\mathrm{S},\pi}$, and get following constraints:

$$\sum_{c \in C_v} \alpha^{\mathrm{W},c,\pi} z_{k,t}^{c} = \gamma_{k,t}^{\mathrm{W},\pi} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T \tag{4.44}$$

$$\gamma_{k,t}^{\mathrm{W},\pi} - \gamma_{k,t-1}^{\mathrm{W},\pi} \leq x_{k,t}^{\mathrm{W}} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T_k' \tag{4.45}$$

$$\gamma_{k,t-1}^{\mathrm{W},\pi} - \gamma_{k,t}^{\mathrm{W},\pi} \leq x_{k,t}^{\mathrm{W}} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T_k' \tag{4.46}$$

$$\sum_{c \in C_v} \alpha^{\mathrm{B},c,\pi} z_{k,t}^{c} = \gamma_{k,t}^{\mathrm{B},\pi} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T \tag{4.47}$$

$$\gamma_{k,t}^{\mathrm{B},\pi} - \gamma_{k,t-1}^{\mathrm{B},\pi} \leq x_{k,t}^{\mathrm{BS}} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T_k' \tag{4.48}$$

$$\gamma_{k,t-1}^{\mathrm{B},\pi} - \gamma_{k,t}^{\mathrm{B},\pi} \leq x_{k,t}^{\mathrm{BS}} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T_k' \tag{4.49}$$

$$\sum_{c \in C_v} \alpha^{\mathrm{S},c,\pi} z_{k,t}^{c} = \gamma_{k,t}^{\mathrm{S},\pi} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T \tag{4.50}$$

$$\gamma_{k,t}^{\mathrm{S},\pi} - \gamma_{k,t-1}^{\mathrm{S},\pi} \leq x_{k,t}^{\mathrm{BS}} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T_k' \tag{4.51}$$

$$\gamma_{k,t-1}^{\mathrm{S},\pi} - \gamma_{k,t}^{\mathrm{S},\pi} \leq x_{k,t}^{\mathrm{BS}} \qquad k \in K_v, \pi \in \Pi_v, v \in V, t \in T_k' \tag{4.52}$$

Constraints (4.44) to (4.45) is equivalent to constraints (4.41), and constraints (4.47) to (4.49) equivalent to (4.42), (4.50) to (4.52) equivalent to constraints (4.43).

4.3.2 Pricing Problem

Reduced cost with the path formulation:

$$\overline{\mathrm{COST}}_2(z_{k,t}) = 0 - u^{(4.3)}_{v_k t} - \sum_{\ell \in L} u^{(4.4)}_{\ell t} \Delta_k \, p^{\mathrm{W}}_{\ell} - \sum_{\ell \in L} u^{(4.5)}_{\ell t} \Delta_k \delta_k \, p^{\mathrm{S}}_{\ell} + \sum_{\ell \in L} \sum_{\ell' \in L \setminus \{\ell\}} u^{(4.6)}_{\ell \ell' t} \Delta_k \, p^{\mathrm{W}}_{\ell'} p^{\mathrm{B}}_{\ell} + \sum_{v \in V_{\mathrm{D}}} \sum_{\ell \in L} u^{(4.7)}_{v \ell t} \Delta_k a^{\mathrm{W}}_{v} p^{\mathrm{B}}_{\ell} - \sum_{\pi \in \Pi_v} u^{(4.44)}_{\pi v_k t} \alpha^{\mathrm{W},\pi} - \sum_{\pi \in \Pi_v} u^{(4.47)}_{\pi v_k t} \alpha^{\mathrm{B},\pi} - \sum_{\pi \in \Pi_v} u^{(4.50)}_{\pi v_k t} \alpha^{\mathrm{S},\pi} \quad (4.53)$$

where v_k is the source node of request k, $u^{(4.3)}_{v_k t} \geq 0$, $u^{(4.4)}_{\ell t} \gtrless 0$, $u^{(4.5)}_{\ell t} \gtrless 0$, $u^{(4.6)}_{\ell \ell'} \geq 0$, $u^{(4.7)}_{v \ell} \geq 0$, $u^{(4.44)}_{\ell v_k t} \gtrless 0$, $u^{(4.47)}_{\ell v_k t} \gtrless 0$ and $u^{(4.50)}_{\ell v_k t} \gtrless 0$ are the values of the dual variables associated with constraints (4.3), (4.4), (4.5), (4.6), (4.7), (4.44), (4.47), (4.50) respectively.

We need to keep constraints (4.28) to (4.31) to linearize the two quadratic terms, and (4.32) to (4.40) to generate a configuration. They also need some addition constraints to define variable $\alpha^{\mathrm{W},\pi}$, $\alpha^{\mathrm{B},\pi}$ and $\alpha^{\mathrm{S},\pi}$. Let p be the path that built by the pricing problem, and p' be the path generated by previous pricing problem. We have to check it against all previously π generated paths:

$$\alpha^{\mathrm{W},\pi} = \bigwedge_{\ell \in L} \left(p^{\mathrm{W}}_{\ell} \equiv {p'}^{\mathrm{W},\pi}_{\ell} \right) \qquad \pi \in \Pi_v \quad (4.54)$$

$$\alpha^{\mathrm{B},\pi} = \bigwedge_{\ell \in L} \left(p^{\mathrm{B}}_{\ell} \equiv {p'}^{\mathrm{B},\pi}_{\ell} \right) \qquad \pi \in \Pi_v \quad (4.55)$$

$$\alpha^{\mathrm{S},\pi} = \bigwedge_{\ell \in L} \left(p^{\mathrm{S}}_{\ell} \equiv {p'}^{\mathrm{S},\pi}_{\ell} \right) \qquad \pi \in \Pi_v \quad (4.56)$$

The above three constraints are highly nonlinear, so after linearization we have:

$$\alpha^{\mathrm{W},\pi} \leq p_\ell^{\mathrm{W}} \cdot p'^{\mathrm{W},\pi}_\ell + (1 - p_\ell^{\mathrm{W}}) \cdot \left(1 - p'^{\mathrm{W},\pi}_\ell\right) \qquad \ell \in L, \pi \in \Pi \tag{4.57}$$

$$\alpha^{\mathrm{W},\pi} + |L| - 1 \geq \sum_{\ell \in L} \left(p_\ell^{\mathrm{W}} \cdot p'^{\mathrm{W},\pi}_\ell + (1 - p_\ell^{\mathrm{W}}) \cdot \left(1 - p'^{\mathrm{W},\pi}_\ell\right)\right) \qquad \pi \in \Pi \tag{4.58}$$

$$\alpha^{\mathrm{B},\pi} \leq p_\ell^{\mathrm{B}} \cdot p'^{\mathrm{B},\pi}_\ell + (1 - p_\ell^{\mathrm{B}}) \cdot \left(1 - p'^{\mathrm{B},\pi}_\ell\right) \qquad \ell \in L, \pi \in \Pi \tag{4.59}$$

$$\alpha^{\mathrm{B},\pi} + |L| - 1 \geq \sum_{\ell \in L} \left(p_\ell^{\mathrm{B}} \cdot p'^{\mathrm{B},\pi}_\ell + (1 - p_\ell^{\mathrm{B}}) \cdot \left(1 - p'^{\mathrm{B},\pi}_\ell\right)\right) \qquad \pi \in \Pi \tag{4.60}$$

$$\alpha^{\mathrm{S},\pi} \leq p_\ell^{\mathrm{S}} \cdot p'^{\mathrm{S},\pi}_\ell + (1 - p_\ell^{\mathrm{S}}) \cdot \left(1 - p'^{\mathrm{S},\pi}_\ell\right) \qquad \ell \in L, \pi \in \Pi \tag{4.61}$$

$$\alpha^{\mathrm{S},\pi} + |L| - 1 \geq \sum_{\ell \in L} \left(p_\ell^{\mathrm{S}} \cdot p'^{\mathrm{S},\pi}_\ell + (1 - p_\ell^{\mathrm{S}}) \cdot \left(1 - p'^{\mathrm{S},\pi}_\ell\right)\right) \qquad \pi \in \Pi \tag{4.62}$$

Notice that, here $p_\ell^{\mathrm{W}}, p_\ell^{\mathrm{B}}, p_\ell^{\mathrm{S}}$ are variables and $p'^{\mathrm{W},\pi}_\ell, p'^{\mathrm{B},\pi}_\ell, p'^{\mathrm{S},\pi}_\ell$ are parameters, so those constraints are linear. Constraints (4.57) and (4.58) are equivalent to constraints (4.54), (4.59) and (4.60) are equivalent to (4.55) and (4.61)(4.62) are equivalent to constraints (4.56). Therefore, the constraints for this model's pricing problem are constraints (4.28)–(4.40) with additional constraints (4.57)–(4.62).

This model is a bit more scalable than Model 1, only if the average number of path originate at one node is smaller than the number of links in the given network. But obversely, the model for pricing problem is more complex. We will compare this model with others in Section 4.6.

4.4 Model 3: Aggregated Traffic & Path Formulation (Reconfiguration Optimized)

The optimization terms is not changed in this model, we still need to minimize both bandwidth requirement and the number of rerouted paths. The difference is that, in this model, we aggregate traffic with same source node. The configuration assignment is based on source node instead of a single request. As the example shown in Fig. 4.3. In time period t_1, there are 20 units of traffic originated on source node v, and configurations C_1, C_2 and C_3 which carry 10, 5 and 5 units of traffic are assigned to node v to provide services to the requests which related to these traffic. We do not care these 20 units of traffic belongs to which re-

quests, and we only need to guarantee that there are configurations to carry these 20 units of traffic. The detection of reconfigurations is achieved by checking every path in the network.

4.4.1 Master Problem

4.4.1.1 Objective

$$\min \mathrm{BW} \quad + \quad \mathrm{PENAL}^{\mathrm{DISRUPT\_B}} \sum_{v \in V} \sum_{t \in T'} \quad x^{\mathrm{BS}}_{v,t} \quad + \quad \mathrm{PENAL}^{\mathrm{DISRUPT\_W}} \sum_{v \in V} \sum_{t \in T'} \quad x^{\mathrm{W}}_{v,t}. \quad (4.63)$$

Same as previous models, BW is the maximum bandwidth requirements among all time periods. $x^{\mathrm{BS}}_{v,t}$ and $x^{\mathrm{W}}_{v,t}$ is used to compute reconfigured traffic, just as what we defined in Section 3.2.4. Since we aggregate traffic, it is hard to detect rerouting for each requests, here we compute reconfigurations for the traffic that originated at each node.

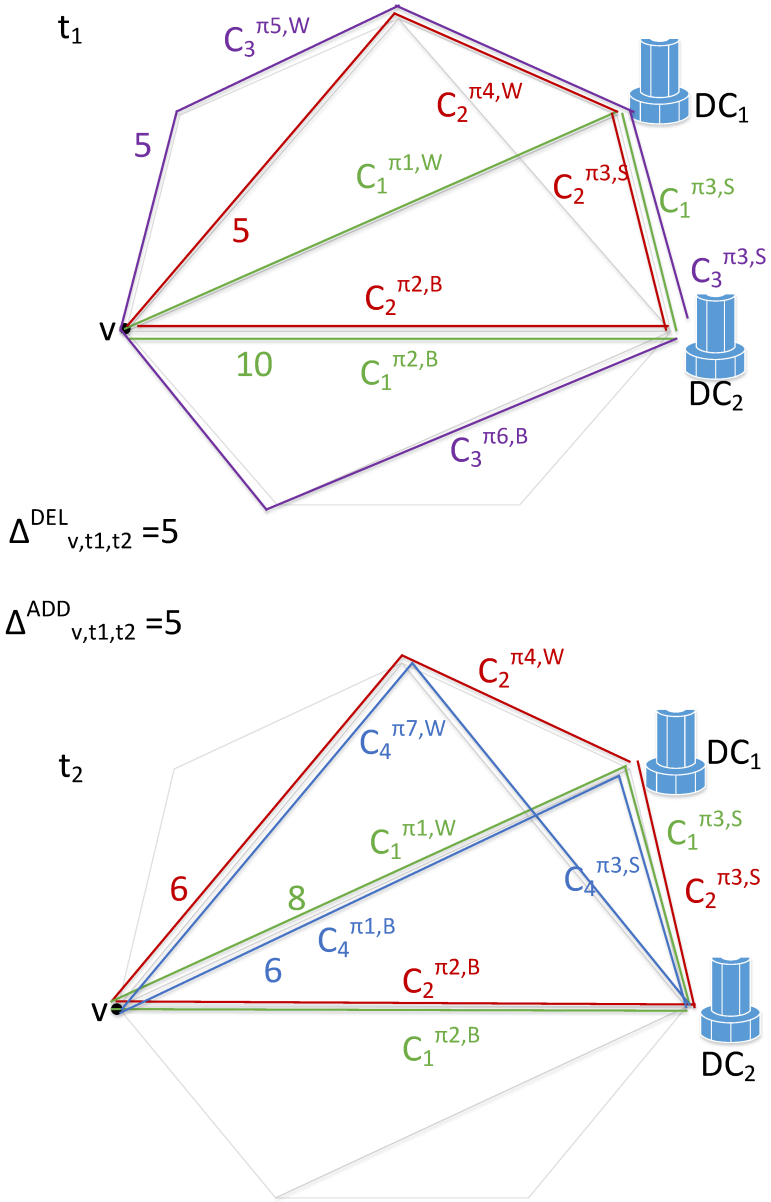

Fig. 4.3: Mapping Example for Model 3

4.4.1.2 Constraints

$$\sum_{c \in C_v} z_t^c \geq \Delta_{v,t} \qquad v \in V, t \in T \tag{4.64}$$

$$\sum_{c \in C} p_\ell^{\mathrm{W},c} z_t^c = \beta_{\ell,t}^{\mathrm{W}} \qquad \ell \in L, t \in T \tag{4.65}$$

$$\sum_{v \in V} \sum_{c \in C_v} \delta_v \, p_\ell^{\mathrm{S},c} z_t^c \; = \beta_{\ell,t}^{\mathrm{S}} \qquad \ell \in L, t \in T \tag{4.66}$$

$$\sum_{c \in C} p_{\ell'}^{\mathrm{W},c} \, p_\ell^{\mathrm{B},c} \, z_t^c \leq \beta_{\ell,t}^{\mathrm{B}} \qquad \ell' \in L, \ell \in L \setminus \{\ell'\}, t \in T \tag{4.67}$$

$$\sum_{c \in C} a_v^{\mathrm{W},c} \, p_\ell^{\mathrm{B},c} \, z_t^c \; \leq \beta_{\ell,t}^{\mathrm{B}} \qquad v \in V_{\mathrm{D}}, \ell \in L, t \in T \tag{4.68}$$

$$\sum_{\pi \in \Pi_v} \; | \sum_{c \in C^{\mathrm{W},\pi}} (z_t^c - z_{t-1}^c)| - (\Delta_{v,t,t-1}^{\mathrm{DEL}} + \Delta_{v,t,t-1}^{\mathrm{ADD}}) \leq 2 x_{v,t}^{\mathrm{W}} \qquad v \in V, t \in T' \tag{4.69}$$

$$\sum_{\pi \in \Pi_v} \sum_{\pi' \in \Pi_v \setminus \pi} \; | \sum_{c \in C^{\mathrm{BS},\pi,\pi'}} (z_t^c - z_{t-1}^c)| - (\Delta_{v,t,t-1}^{\mathrm{DEL}} + \Delta_{v,t,t-1}^{\mathrm{ADD}}) \leq 2 x_{v,t}^{\mathrm{BS}} \qquad v \in V, t \in T' \tag{4.70}$$

$$\mathrm{BW} \geq \sum_{\ell \in L} (\beta_{\ell,t}^{\mathrm{W}} + \beta_{\ell,t}^{\mathrm{B}} + \beta_{\ell,t}^{\mathrm{S}}) \cdot \|\ell\| \qquad t \in T \tag{4.71}$$

$$z_t^c \in \mathbb{R} \qquad c \in C, t \in T \tag{4.72}$$

$$\mathrm{BW}, \beta_{\ell,t}^{\mathrm{W}}, \; \beta_{\ell,t}^{\mathrm{B}}, \; \beta_{\ell,t}^{\mathrm{S}} \in \mathbb{R} \qquad \ell \in L, t \in T \tag{4.73}$$

Constraints (4.64) guarantee that the bandwidth of configurations that assigned on the node v is larger than the total bandwidth requirements of all requests originated on v, in other word, all the bandwidth requirements will be satisfied. Just as in Model 1 and Model 2. Constraints (4.65) and (4.66) compute the working and synchronization bandwidth requirements on link ℓ at time period t, respectively. Constraints (4.67) ensure sufficient shared backup bandwidth requirements on link ℓ at time period t subject to a single link failure. Constraints (4.68) guarantee sufficient backup bandwidth ℓ to handle any data center failure. The last two set of constraints, i.e., (4.72) and (4.73), define the domain of the variables.

Constraints (4.69) and (4.70) are constraints about reconfigurations. Traffic variation on source node v between time period t and $t-1$ can be expressed as $\Delta_{v,t,t-1}^{\mathrm{DEL}} + \Delta_{v,t,t-1}^{\mathrm{ADD}}$. On

a single working path π, bandwidth change is $|\sum_{c\in C^{\mathrm{W},\pi}}(z_t^c - z_{t-1}^c)|$ and $|\sum_{c\in C^{\mathrm{BS},\pi,\pi'}}(z_t^c - z_{t-1}^c)|$ represent the bandwidth change on path π and π' where π is used as a backup path and π' is used as a synchronization path. Therefore, reconfigured bandwidth on source node v at time period t is $\frac{1}{2}(\sum_{\pi\in\Pi_v}|\sum_{c\in C^{\mathrm{W},\pi}}(z_t^c - z_{t-1}^c)| - (\Delta_{v,t,t-1}^{\mathrm{DEL}} + \Delta_{v,t,t-1}^{\mathrm{ADD}}))$ for working paths or $\frac{1}{2}(\sum_{\pi\in\Pi_v}\sum_{\pi'\in\Pi_v\setminus\pi}|\sum_{c\in C^{\mathrm{BS},\pi,\pi'}}(z_t^c - z_{t-1}^c)| - (\Delta_{v,t,t-1}^{\mathrm{DEL}} + \Delta_{v,t,t-1}^{\mathrm{ADD}}))$ for backup paths.

As example shown in Fig. 4.3. When time period change from t_1 to t_2, there are 5 traffic units are deleted and 5 units are new. Since we do not care about traffic details, the removed traffic and new traffic can be anyone. The graph shows part of mapping result, and it satisfied conditions in scenario 3. The traffic variation is 10 units (5 deleted, 5 new), and if we concentrate on working path variation, bandwidth difference can be computed by $2(\pi_1)+1(\pi_4)+5(\pi_5)+6(\pi_7)=14$. 10 of the 14 units are caused by the change of traffic, and 4 of them are because of reconfiguration. Assuming we remove two units from configuration c_1, and add one to c_2 and another one to c_4. Indeed, there are only 2 units of traffic being reconfigured, and the left part of constraints (4.69) and (4.70) will compute these changes twice (once for decrease on one configuration, and once for increase on another configuration), that's where $\frac{1}{2}$ comes from.

In *Scenario I*, requests cannot be reconfigured, so we add following constraints:to define the value of $x_{v,t}^{\mathrm{W}}, x_{v,t}^{\mathrm{BS}}$:

$$x_{v,t}^{\mathrm{W}} = x_{v,t}^{\mathrm{BS}} = 0 \quad v\in V, t\in T' \tag{4.74}$$

By forcing these two variables equal to zero, we force all of working paths, backup paths and synchronization paths must keep same for all time periods.

In *Scenario II*, we add additional constraints as follows:

$$x_{v,t}^{\mathrm{W}} = 0 \quad v\in V, t\in T' \tag{4.75}$$

$$x_{v,t}^{\mathrm{BS}} \in \mathbb{R} \quad v\in V, t\in T' \tag{4.76}$$

In this scenario, only backup and synchronization paths can be modified. These additional constraints force working paths keep same, and calculate the reconfigurations for backup paths and synchronization paths.

In *Scenario III*, reconfiguration is total free, so we add:

$$x^{\mathrm{W}}_{v,t}, x^{\mathrm{BS}}_{v,t} \in \mathbb{R} \quad v \in V, t \in T' \tag{4.77}$$

In this model, the number of constraints will not be influenced by the number of service requests. But the main issues with it is the linearization of constraints (4.69) and (4.70). There will be $2^{|\Pi|}$ number of constraints to linearize constraints (4.69) and $2^{2|\Pi|}$ constraints to linearize (4.69). Therefore solving master problem will be really costly with the fact that the size of paths set Π is keeping increasing.

4.5 Model 4: Aggregated Traffic & Path Formulation

In this model, the purpose is slightly different from previous models. Here, we only minimize the bandwidth requirement, but not the number of rerouted paths. The traffic is also aggregated in order to keep the advantages of the scalability. The assumption of which traffic unit can be reconfigured is different from previous models. Here, we concentrate on the amount of traffic based on each node. If $\Delta_{v,t} - \Delta_{v,t-1} \geq 0$, we assume that there are only $(\Delta_{v,t} - \Delta_{v,t-1})$ units new traffic and none of traffic is deleted at the end of time period $t-1$. Oppositely, if $\Delta_{v,t} - \Delta_{v,t-1} < 0$, we assume that there are only $(\Delta_{v,t-1} - \Delta_{v,t})$ units traffic is removed and no new traffic is added in. If we look at the example in Fig. 4.3, there are actually 5 units traffic deleted and 5 units new traffic. But in this model, we treat it as there is no traffic changes (no deletion and no creation).

4.5.1 Master Problem

In the master problem, multiple configurations will be assigned to requests on same source node. It can be understood by aggregated all requests with same source node as one, and this one can choose more than one configurations. The goal of master problem is to minimize the summation of the bandwidth requirements of all selected configurations, but do not take care of optimizing reconfigurations.

4.5.1.1 Objective

$$\min \max_{t \in T} \sum_{\ell \in L} (\beta^{\mathrm{W}}_{\ell,t} + \beta^{\mathrm{B}}_{\ell,t} + \beta^{\mathrm{S}}_{\ell,t}) \cdot \|\ell\| \,. \tag{4.78}$$

where $\|\ell\|$ being the length of link ℓ. The object function is to minimize the overall bandwidth requirements which should be no smaller than the bandwidth requirements at any time period. $\max\limits_{t \in T} \sum\limits_{\ell \in L} (\beta^{\mathrm{W}}_{\ell,t} + \beta^{\mathrm{B}}_{\ell,t} + \beta^{\mathrm{S}}_{\ell,t}) \cdot \|\ell\|$ is equivalent to BW in previous models.

4.5.1.2 Constraints

We keep constraints (4.64)–(4.73) in this model, but do not need constraints (4.69) and (4.70) anymore.

In *Scenario I*, requests cannot be reconfigured, so we add some constraints. For every $v \in V, \pi \in \Pi_v, t \in T'$:

$$\sum_{c \in C^{\mathrm{W},\pi}} (z^c_t - z^c_{t-1}) \begin{cases} \leq 0 & \text{if } \Delta_{v,t} < \Delta_{v,t-1} \\ \geq 0 & \text{if } \Delta_{v,t} \geq \Delta_{v,t-1} \end{cases} \tag{4.79}$$

$$\sum_{c \in C^{\mathrm{B},\pi}} (z^c_t - z^c_{t-1}) \begin{cases} \leq 0 & \text{if } \Delta_{v,t} < \Delta_{v,t-1} \\ \geq 0 & \text{if } \Delta_{v,t} \geq \Delta_{v,t-1} \end{cases} \tag{4.80}$$

$$\sum_{c \in C^{\mathrm{S},\pi}} (z^c_t - z^c_{t-1}) \begin{cases} \leq 0 & \text{if } \Delta_{v,t} < \Delta_{v,t-1} \\ \geq 0 & \text{if } \Delta_{v,t} \geq \Delta_{v,t-1} \end{cases} \tag{4.81}$$

Constraints (4.79) check every path to force the working paths not to change between one time period to previous time period. Since we aggregate all the requests originated on same node, it is hard to check configuration changes for every request. Here, we switch to another strategy. If traffic increase between time period $t-1$ and t, and there is no configuration's bandwidth decrease, we consider this situation as no reconfigurations. Otherwise, we consider it as configuration changes. Therefore, constraints (4.79) guarantee that the bandwidth of each working path which originated on source node v could only increase or keep same when traffic demand is increasing on node v during time period t and $t-1$, and could only decrease or not change if traffic demand is decreasing. By doing this, we forth the 'legacy' requests to use same working path as previously.

Constraints (4.80) and (4.81) force the backup paths and synchronization paths not to be reconfigured, respectively.

In *Scenario II*, only backup and synchronization paths can be modified. So, in scenario 2 we add constraints (4.79). And in *Scenario III*, all the paths are free to change, nothing need to be added.

As same as the reason we explained in Section 4.2.1.2 we do not want three different pricing problems, so that $\gamma_t^{*,\pi}$ is introduced to represent variable $\sum_{c \in C^{*,\pi}} z_t^c$. Therefore, constraints (4.79) - (4.81) can be rewritten as:

$$\sum_{c \in C^{\mathrm{W},\pi}} z_t^c = \gamma_t^{\mathrm{W},\pi} \qquad \pi \in \Pi, t \in T \tag{4.82}$$

$$\sum_{c \in C^{\mathrm{B},\pi}} z_t^c = \gamma_t^{\mathrm{B},\pi} \qquad \pi \in \Pi, t \in T \tag{4.83}$$

$$\sum_{c \in C^{\mathrm{S},\pi}} z_t^c = \gamma_t^{\mathrm{S},\pi} \qquad \pi \in \Pi, t \in T \tag{4.84}$$

$$\gamma_t^{\mathrm{W},\pi} - \gamma_{t-1}^{\mathrm{W},\pi} \begin{cases} \leq 0 & \text{if } \Delta_{v,t} < \Delta_{v,t-1} \\ \geq 0 & \text{if } \Delta_{v,t} \geq \Delta_{v,t-1} \end{cases} \tag{4.85}$$

$$\gamma_t^{\mathrm{B},\pi} - \gamma_{t-1}^{\mathrm{B},\pi} \begin{cases} \leq 0 & \text{if } \Delta_{v,t} < \Delta_{v,t-1} \\ \geq 0 & \text{if } \Delta_{v,t} \geq \Delta_{v,t-1} \end{cases} \tag{4.86}$$

$$\gamma_t^{\mathrm{S},\pi} - \gamma_{t-1}^{\mathrm{S},\pi} \begin{cases} \leq 0 & \text{if } \Delta_{v,t} < \Delta_{v,t-1} \\ \geq 0 & \text{if } \Delta_{v,t} \geq \Delta_{v,t-1} \end{cases} \tag{4.87}$$

4.5.2 Pricing Problem

Recall that the pricing problem (PP) will determine augmenting configurations, i.e., routes for W, B and S paths such that their addition to the restricted master problem will entail an improvement of the optimal value of the current restricted master. Each PP is written for a given source node v and for a given time period t. Parameters δ_v retain their definition for a node v as in the RMP.

The objective of $\mathrm{PP}_4(v,t)$ with $v \in V, t \in T$ is to minimize the reduced cost $\overline{\mathrm{COST}}^4(z_{v,t})$ as

obtained from the RMP, defined as:

$$\overline{\text{COST}}_4(z_{v,t}) = 0 - u_{vt}^{(4.64)} - \sum_{\ell \in L} u_{\ell t}^{(4.65)} p_\ell^{\text{W}} - \sum_{\ell \in L} u_{\ell t}^{(4.66)} \delta_v \, p_\ell^{\text{S}} + \sum_{\ell \in L} \sum_{\ell' \in L \setminus \{\ell\}} u_{\ell \ell' t}^{(4.67)} p_{\ell'}^{\text{W}} p_\ell^{\text{B}} + \sum_{v \in V_{\text{D}}} \sum_{\ell \in L} u_{v \ell t}^{(4.68)} a_v^{\text{W}} p_\ell^{\text{B}} + \sum_{\pi \in \Pi} u_{\pi t}^{(4.85)} + \sum_{\pi \in \Pi} u_{\pi t}^{(4.86)} + \sum_{\pi \in \Pi} u_{\pi t}^{(4.87)} \quad (4.88)$$

$u_{vt}^{(4.64)} \geq 0$, $u_{\ell t}^{(4.65)} \gtreqless 0$, $u_{\ell t}^{(4.66)} \gtreqless 0$, $u_{\ell \ell' }^{(4.67)} \geq 0$, $u_{v \ell}^{(4.68)} \geq 0$ $u_{\pi t}^{(4.85)} \gtreqless 0$, $u_{\pi t}^{(4.86)} \gtreqless 0$, $u_{\pi t}^{(4.87)} \gtreqless 0$ are the values of the dual variables associated with constraints (4.64), (4.65), (4.66), (4.67), (4.68), (4.85), (4.86), (4.87) respectively. The first explicit 0 term stems from the RMP objective, which does not contain the configuration variable z_t^c.

The constraints of PP are used to generate a configuration which is same for each model, and we could reuse constraints (4.32) to (4.40) in this model.

There are two quadratic terms $p_{\ell'}^{\text{W}} p_\ell^{\text{B}}$ and $a_v^{\text{W}} p_\ell^{\text{B}}$ which can be easily linearized through the introduction of two sets of binary variables $p_{\ell \ell'}^{\text{WB}}$ and $p_{v \ell}^{\text{WB}}$ just as in Model 1 (see Section 4.2.2), and the constraints we need to use are as same as constraints (4.28)–(4.31).

4.6 Comparison of Models

We compare the scalability of all four models proposed in this chapter, and summarize their advantages and disadvantages in this section.

The first two models are non aggregated traffic models, they can calculate the exact number of rerouting paths. But these two models have scalability issues. The one with link formulation (Model 1) has a large number of constraints in the master. Constraints (4.3) gives $|K| \times |T|$ columns. Constraints (4.4) and (4.5) gives $|L| \times |T|$ columns, respectively. $|L| \times |L| \times |T|$ columns for constraints (4.6), $|V_{\text{D}}| \times |L| \times |T|$ columns for constraints (4.7). And after linearization, each one of the constraints (4.18)–(4.26) contributes $|K| \times |L| \times |T|$ columns. Therefore, we could predict that it is time consuming if we want to solve a problem with a big size of service requests and a network contains large number of links. In fact, we did the experiments with a 42-links network by using 4 six-core processors running 2.667

GHz. The running time exceed 50 hours when the number of service requests is more than 100.

Compared with the Model 1, the constraints for master problem of Model 2 (the one with path formulation) is slightly simpler (in general). In Model 1, we need $9 \times |K| \times |L| \times |T|$ columns due to constraints (4.18)–(4.26). In Model 2, we only have approximately $9 \times |K| \times |\Pi_{\text{AVG}}| \times |T|$ columns for the constraints (4.44)–(4.52) ($|K| \times |\Pi_{\text{AVG}}| \times |T|$ for each). $|\Pi_{\text{AVG}}|$ is the average number of paths originating at one node. Although this parameter really depends on the topology of the network, in general, it is much smaller than the number of links.

But while we end up with a more competitive model for master problem, we also get a more complex pricing problem. We could see Section 4.3.2 that constraints (4.57),(4.59) and (4.61) generate $3 \times |L| \times |\Pi|$ columns. Since we need to solving almost $|K| \times |T|$ times pricing problem after one iteration of master problem, we do not expect a heavy pricing problem.

Apart from the issue caused by model in PP, actually, the model of MP is still not efficient for large datasets. Since the number of constraints will increase as the number of requests increase (in practice, the number of requests can be really huge), when we solve a problem with a large number of requests, this model is not scalable enough.

In Model 3, since we aggregated traffic, the number of constraints will not be affected by number of requests. But it is hard to find an efficient way to linearize constraints (4.69) and (4.70). Assuming there are about $|\Pi_{\text{AVG}}|$ different paths originated on each node, and let $\sum_{c \in C^{\text{W},\pi}} (z_t^c - z_{t-1}^c) = \text{A}_\pi$. The first part of constraints (4.69), $\sum_{\pi \in \Pi_v} | \sum_{c \in C^{\text{W},\pi}} (z_t^c - z_{t-1}^c)|$ can be linearized as:

$$\pm \text{A}_{\pi_1} \pm \text{A}_{\pi_2} \pm \text{A}_{\pi_3} \pm \cdots\cdots \tag{4.89}$$

There will be approximately $2^{|\Pi_{\text{AVG}}|}$ constraints for (4.69) after linearization and $2^{2|\Pi_{\text{AVG}}|}$ constraints for (4.70). That means after linearization, we have $|V| \times |T| \times 2^{|\Pi_{\text{AVG}}|} + |V| \times |T| \times 2^{2|\Pi_{\text{AVG}}|}$ columns only for counting reconfigurations, which is too costly if we only use column generation (considering the fact that the size of paths set is keeping increasing after each time a PP is solved). This problem could be solved by involving an additional technology called "row generation ", which will be left as future work.

The last model (Model 4) is the most scalable one. Benefit from aggregating traffic, it

can deal with a large amount of service requests. Due to the path formulation, it is also adapted to networks with a large set of links. The drawbacks of this model are: 1) it is not accurate when it detects new requests (some requests will be considered as "legacy "one and be forced to use the legacy configurations in scenario 1 and 2), 2) it has no capability to optimize rerouting paths.

The summarized comparison of these four models is shown in tables below. Table 4.1 and Table 4.2 describe the number of variables and constraints used in each model, respectively. Table 4.3 list the comparison of pricing problems for each model.

However, after comparing the scalability of these four models and analyzing the drawbacks of each model, we believe that the Model 4 offers the best compromise with respect to scalability vs. accuracy. The less accurate of the problem is not a really big issue as long as all the requests can be satisfied, especially when we can get the accurate solution for scenario 3. As a consequence, we only implemented Model 4.

Master Problems	Variables: #(Numbers)		
Model 1	$\beta^{\mathrm{W}}_{\ell,t}$: $\#(\lvert L\rvert \times \lvert T\rvert)$ $\beta^{\mathrm{B}}_{\ell,t}$: $\#(\lvert L\rvert \times \lvert T\rvert)$ $\beta^{\mathrm{S}}_{\ell,t}$: $\#(\lvert L\rvert \times \lvert T\rvert)$	$z^{c}_{k,t}$: $\#(\lvert K\rvert \times \lvert T\rvert \times \lvert C\rvert)$	
Model 2		$x^{\mathrm{W}}_{k,t}$: $\#(\lvert K\rvert \times \lvert T\rvert)$ $x^{\mathrm{BS}}_{k,t}$: $\#(\lvert K\rvert \times \lvert T\rvert)$	$\alpha^{\mathrm{W},c,\pi}$: $\#(\lvert C\rvert \times \lvert \Pi\rvert)$ $\alpha^{\mathrm{B},c,\pi}$: $\#(\lvert C\rvert \times \lvert \Pi\rvert)$ $\alpha^{\mathrm{S},c,\pi}$: $\#(\lvert C\rvert \times \lvert \Pi\rvert)$
Model 3			$x^{\mathrm{W}}_{v,t}$: $\#(\lvert V\rvert \times \lvert T\rvert)$ $x^{\mathrm{BS}}_{v,t}$: $\#(\lvert V\rvert \times \lvert T\rvert)$
Model 4		z^{c}_{t}: $\#(\lvert C\rvert \times \lvert T\rvert)$	

Table 4.1: Comparison of number of variables used for master problems in each model

Master Problems	#Numbers (Constraints)	
Model 1	$\#\lvert K\rvert \times \lvert T\rvert$ (4.3) $\#\lvert L\rvert \times \lvert T\rvert$ (4.4), (4.5) $\#\lvert L\rvert \times \lvert L\rvert \times \lvert T\rvert$ (4.6)	$\#\lvert K\rvert \times \lvert L\rvert \times \lvert T\rvert$ (4.18)–(4.26)
Model 2	$\#\lvert V_{\mathrm{D}}\rvert \times \lvert L\rvert \times \lvert T\rvert$ (4.7) $\#\lvert T\rvert$ (4.11)	$\#\lvert K\rvert \times \lvert \Pi\rvert \times \lvert T\rvert$ (4.44)–(4.52)
Model 3	$\#\lvert V\rvert \times \lvert T\rvert$ (4.64) $\#\lvert L\rvert \times \lvert T\rvert$ (4.65), (4.66)	$\#\lvert V\rvert \times \lvert T\rvert \times 2^{\lvert \Pi_{\mathrm{AVG}}\rvert}$ (4.69) $\#\lvert V\rvert \times \lvert T\rvert \times 2^{2\lvert \Pi_{\mathrm{AVG}}\rvert}$ (4.70)
Model 4	$\#\lvert L\rvert \times \lvert L\rvert \times \lvert T\rvert$ (4.67) $\#\lvert V_{\mathrm{D}}\rvert \times \lvert L\rvert \times \lvert T\rvert$ (4.68) $\#\lvert T\rvert$ (4.71)	$\#\lvert \Pi\rvert \times \lvert T\rvert$ (4.82)–(4.87)

Table 4.2: Comparison of number of constraints used for master problems in each model

Pricing	Variables: #(Numbers)		#Numbers (Constraints)	
Model 1	p_ℓ^{W}: $\#(\lvert L\rvert)$ p_ℓ^{B}: $\#(\lvert L\rvert)$			
Model 2	p_ℓ^{S}: $\#(\lvert L\rvert)$ a_v^{W}: $\#(\lvert V\rvert)$ a_v^{B}: $\#(\lvert V\rvert)$ d_v^{W}: $\#(\lvert V\rvert)$	$\alpha^{\mathrm{W},\pi}$: $\#(\lvert \Pi\rvert)$ $\alpha^{\mathrm{B},\pi}$: $\#(\lvert \Pi\rvert)$ $\alpha^{\mathrm{S},\pi}$: $\#(\lvert \Pi\rvert)$	$\#\lvert V\rvert$ (4.32)–(4.34) $\#\lvert L\rvert$ (4.35) #1 (4.36) $\#\lvert V_{\mathrm{D}}\rvert$ (4.37)	$\#\lvert L\rvert \times \lvert \Pi\rvert$ (4.57),(4.59),(4.61) $\#\lvert \Pi\rvert$ (4.58),(4.60),(4.62)
Model 3	d_v^{B}: $\#(\lvert V\rvert)$ d_v^{S}: $\#(\lvert V\rvert)$		$\#\lvert L\rvert \times \lvert L\rvert$ (4.28),(4.29) $\#\lvert L\rvert \times \lvert V_{\mathrm{D}}\rvert$ (4.30),(4.31)	
Model 4	$p_{\ell\ell'}^{\mathrm{WB}}$: $\#(\lvert L\rvert \times \lvert L\rvert)$ $p_{v\ell}^{\mathrm{WB}}$: $\#(\lvert L\rvert \times \lvert V\rvert)$			

Table 4.3: Comparison of number of variables and constraints used for pricing problems in each model

Chapter 5

Solution Process

5.1 Strategy with Parallel Computing

As mentioned in previous chapter, the problem could be really large. Therefore, in order to save CPU time, we implemented this problem by using parallel computing with MPJ. Here we only describe the solution process for Model 4. The purpose of master problem is to get an optimal solution based on the whole network and all the service requests, and constraints in the master problem are highly related. Therefore, we only solved pricing problem in parallel but not for RMP.

As shown in Fig. 5.1, we create a process for each source node, and each processor response for solving the pricing problem associated with that node. The solution procedure is described as follows:

- Create one processor, call it master processor, to solve master problem. And send the dual values to other processors, call them sub-processors.
- Every sub-processor solves one pricing problem PP(v, t) associated with a single node v and a single time period t.
- If PP(v, t) get a negative reduced cost, then the sub-processor send the generated configuration to master processor.

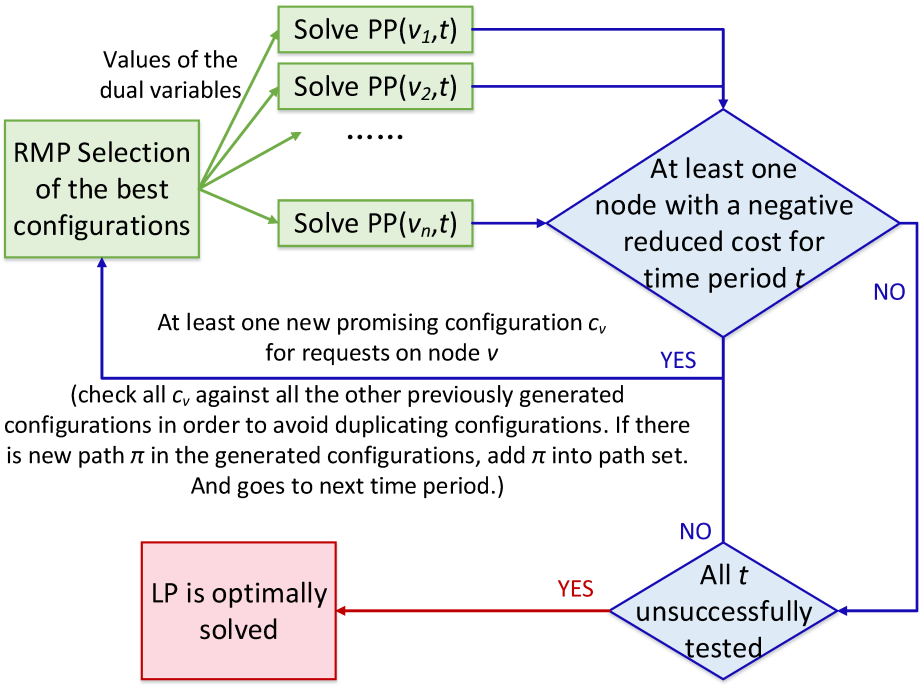

Fig. 5.1: Flowchart for Column Generation in Parallel.

- Master processor collects all these configurations, and adds them to the configuration set. If there are new paths, then they will be added into the set of paths.
- If the iteration does not exceed the set of time periods, in each iteration t_1 will be updated to t_2, t_2 will be updated to t_3 and so on so forth, otherwise $t = t_0$. When there are no negative result got by all sub-processors for all $t \in T$, then we reach the stop condition.
- Get an optimal LP solution for the problem.

Here we introduce another conception 'round' in order to better explain the advantages and drawbacks of this strategy. PP solved for a round defined as all PP(v, t) $(v \in V, t \in T)$ solved for once. By implementing the strategy described above, we solve all PP$(v, t), v \in V$ in parallel for a given time period t, therefore we solve $|V|$ pricing problems with same set of dual values for one iteration.

The advantage of this strategy is the parallelization of pricing problems solving. It solves PP for all the nodes and can get several new columns in one iteration. The drawback is that all the PP$(v,t)(v \in V$, t is given) in a round are solved with the same set of dual values, if the configuration which got by PP(v_1,t_1) can be updated into configuration set before PP(v_2,t_1) is solving, with the parallel strategy, the dual values can not have much improvement. While the dual values which computed by RMP with new set of configurations may help PP(v_2,t_1) to get a better configuration.

Compared with the so-called 'round robin' strategy, this one will have slightly more number of rounds (when complete the whole procedure in 'round robin', then it calls a 'round' in parallel) and much larger memory requirements, but save a lot CPU time (more details in Section 6.3).

5.2 Other Unexplored Strategies

Since there are some drawbacks for the parallel strategy we described above, it could be improved. We thought about two different strategies which are explained below.

5.2.1 An Improved Serial Strategy

Different from the round robin strategy, here we select the node v that will contribute most to the improvement for master problem, which means the reduced cost of PP(v,t) for a given time period t is the most negative one based on the current set of dual variables. We can expect that there will be less iterations for MP to get the optimal solution. However, it does not change the stopping condition which is quite time consuming. We still need a complete round (when such a round is activated) for all time periods with no negative reduced cost.

To overcome the drawback, we can use a large PP that embed all source nodes with traffic. Instead of solving PP(v,t) for every $v \in V$, we generate new configuration by using PP(t) which is response for finding the 'best' node v and generate a configuration originating on node v. The flowchart is shown in Fig. 5.2.

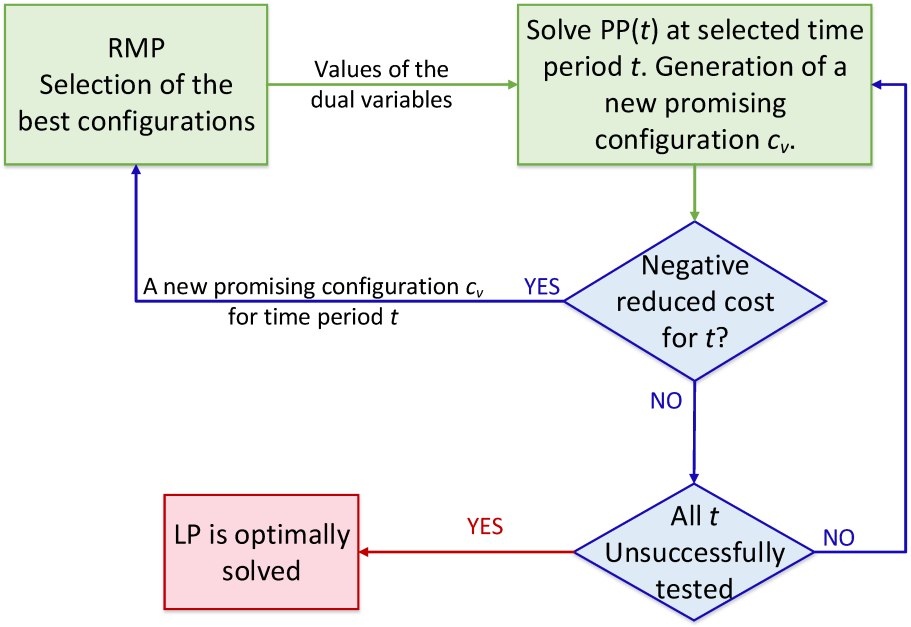

Fig. 5.2: Flowchart for the Improved Serial Strategy.

By using the large PP, we change the stopping condition and do not need such a complete round to reach the stopping condition. But it is still difficult to beat the parallel strategy which described in Section 5.1 with the comparison of overall CPU time, since the PP is so large and could only provide a single configuration at one time.

5.2.2 An Improved Parallel Strategy

This strategy is a kind of balance between the two extreme strategies, round robin and all pricing problems are solved in parallel. Here, we select a subset of PP(v, t) to be solved in parallel for each iteration. There are two schemes to choose such a subset.

- Divide the source nodes into groups, and grouping them in a arbitrary way. Since the traffic generation is random (see Chapter 6), which means the service requests originating on each node are different and transformable, the advantages of this scheme is very limited.

- Select the subset base on the lower bound and upper bound of the reduced cost. The table below is a small example for source nodes $v \in V$ at time period t. Nodes with a positive lower bound like node v_2 will be discarded. Because solely solving $\mathrm{PP}(v_2, t)$ can not find an augmenting configuration. The nodes like v_3 could be our best choice, and the smaller lower bound means it's more possible to find a configuration that will improve more on master problem. Therefore, the flowchart will be very similar with Fig. 5.3. When implement this scheme, we need to be careful with the computational cost for the lower bound and upper bound, otherwise, it may lose the advantages with CPU time.

node	v_1	v_2	v_3	...	v_n
LB	-3	2	-6		-7
UB	3	11	-1		1

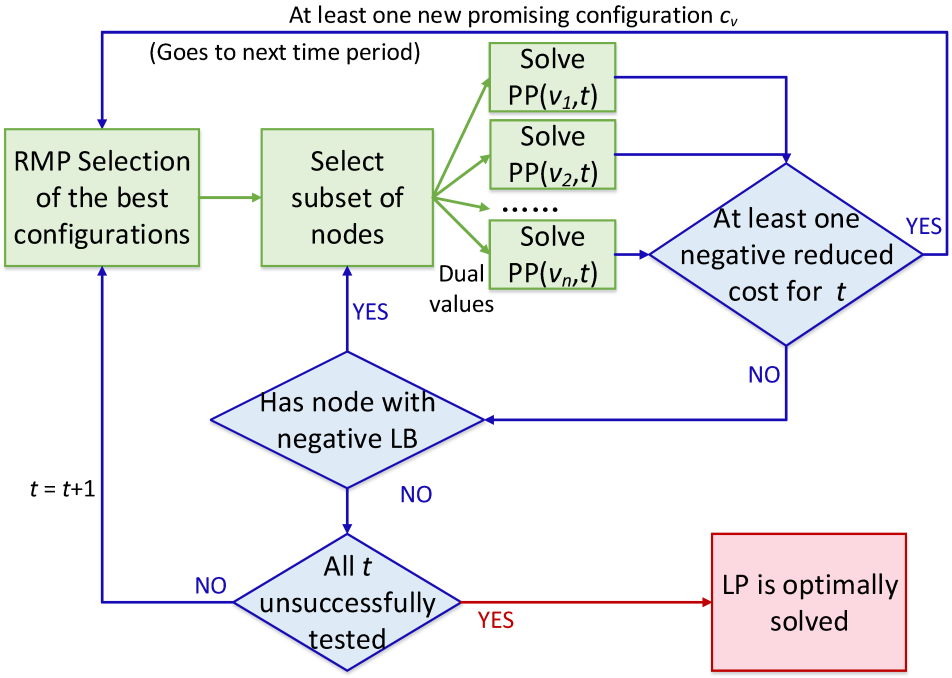

Fig. 5.3: Flowchart for the Improved Parallel Strategy.

Chapter 6

Numerical Results

The optimization models were implemented and tested on several data sets. We first describe the experiment data sets in Section 6.1. Next, we present the numerical results of Model 4 (described in Section 4.5) in Section 6.2. The comparison for saving the problem with parallel strategy and serial strategy is presented in Section 6.3.

6.1 Data Sets

In this section, we first describe the network we use and the location of data centers in this network. Then we will introduce how we divided the network into different regions. The traffic patterns is displayed next, and we will summarize the four data sets we used in our experiments by a table at last.

6.1.1 Network and Location of Data Centers

We use a 24-node USA network with 43 no-direction links (as illustrated in Fig. 6.1 and Fig. 6.2) , which will be divided into different regions, each with their own traffic pattern.

We have two different data center sets, and in each set there are 4 data centers.

- *DC #1*: We assume the 4 data centers located in UT (node 7), NM(node 10), IL (node 11), MS (node 17) as illustrated in Fig. 6.1.

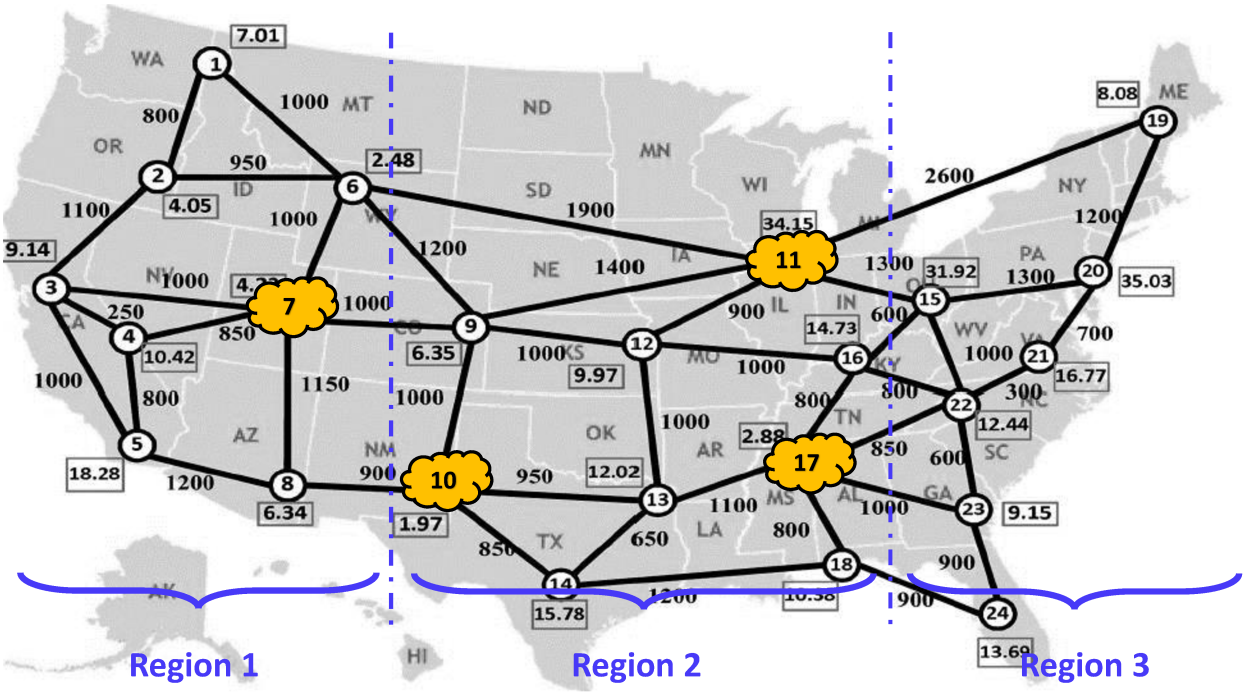

Fig. 6.1: USA Network with 3 Regions and DC #1.

- *DC #2*: We choose nodes located in CA (node 3), CO(node 9), KY (node 16), VA (node 21) as data centers (see Fig. 6.2). .

6.1.2 Time Periods and Regions

We use statistics illustrated in Fig. 6.3, which is calculated by Weibo, as a reference of traffic distribution during 24 hours in our experiments. "Weibo is a Chinese microblogging (weibo) website. Akin to a hybrid of Twitter and Facebook, it is one of the most popular sites in China, in use by well over 30% of Internet users, with a market penetration similar to the United States' Twitter" [48]. The statistics is about the number of times that a top Chinese word "weibo" (which is also the name of the website) was mentioned in the text presented on this website by their clients in 24 hours. Therefore, the traffic distribution for every 8 hours in a day is summarized in Table 6.1.

We define two region sets and separate 24 hours in a day into three time periods.

- *RS #1*: The network is divided into 3 regions, as illustrated in Fig. 6.1, and the traffic distribution in each region is based on the number of servers (nodes) in that region as shown

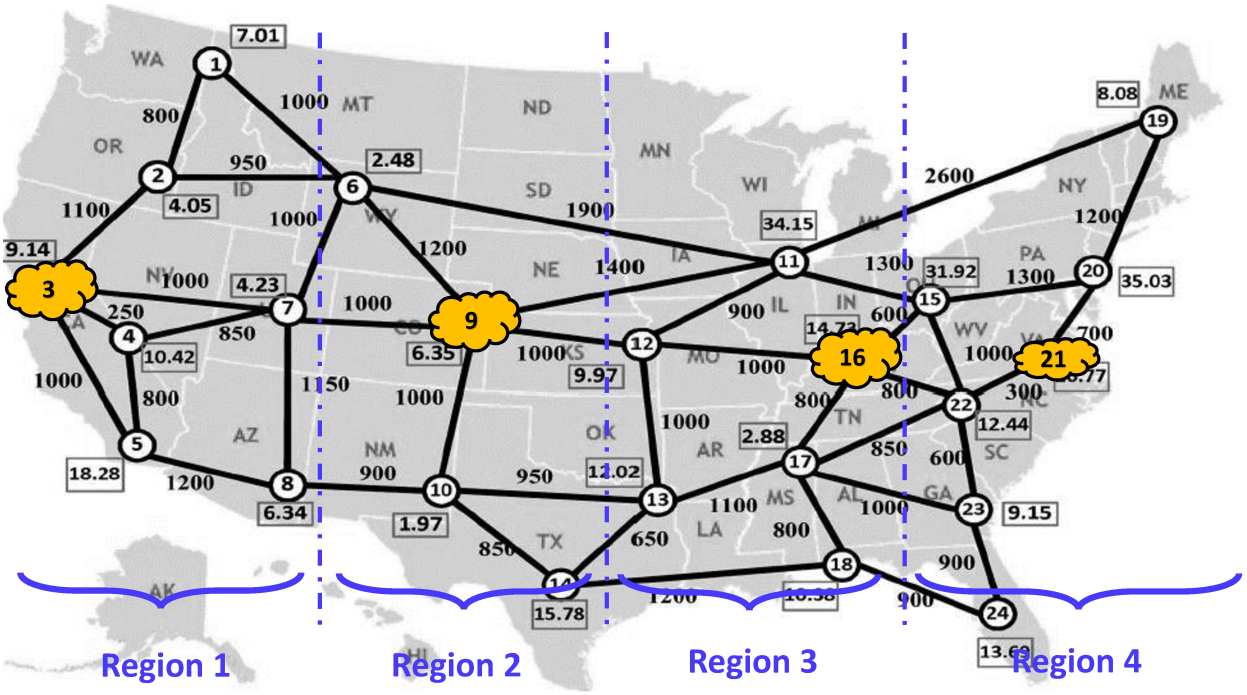

Fig. 6.2: USA Network with 4 Regions and DC #2.

	8am - 4pm	4pm - 12am	12am - 8am
Traffic volume %	48.41%	37.94%	13.64%

Table 6.1: Traffic distribution for every 8 hours in a day

in Table 6.2. The relationship between regions and time periods is described in Table 6.4.

- *RS #2*: The network is divided into 4 regions, as illustrated in Fig. 6.2, and traffic volume in regions is shown in Table 6.3. The relationship between regions and time periods is described in Table 6.5. .

	Region 1	Region 2	Region 3
Traffic volume %	33.33 %	37.5 %	29.17 %

Table 6.2: Traffic distribution in each region for RS #1

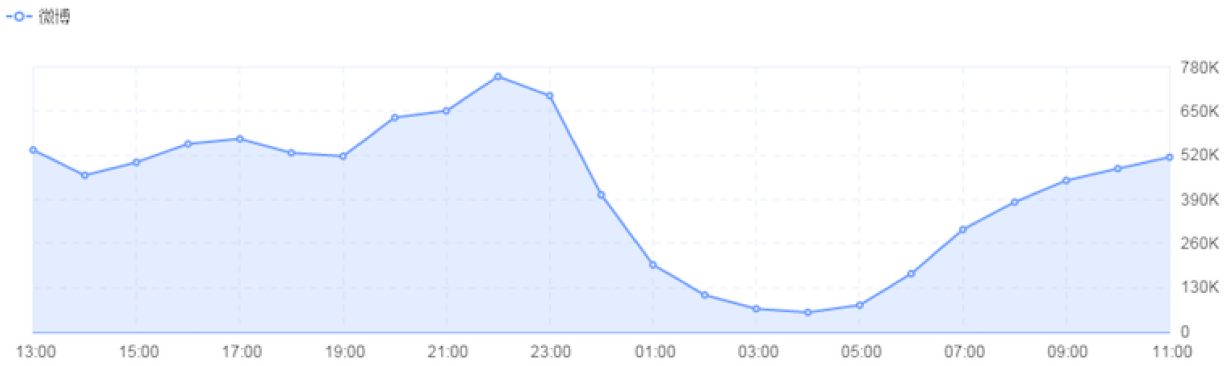

Fig. 6.3: Heated Debate of the Word "Weibo "in 24 Hours.

	Region 1	Region 2	Region 3	Region 4
Traffic volume %	29.17 %	16.67%	25%	29.17 %

Table 6.3: Traffic distribution in each region for RS #2

6.1.3 Traffic Patterns

Three traffic pattern are been considered in our experiments.

- *Pattern #1*: 20% of traffic volume in each time period of a region just last two periods, while the other 80% last just for the single time period.

- *Pattern #2*: 80% of traffic in each time period last two slots, 20% last just one.. .

- *Pattern #3*: 50% of traffic in each time period last two slots, 50% last just one.. .

Time Periods	Region 1	Region 2	Region 3
t1	4pm - 12am (37.94%)	12am - 8am (13.64%)	8am - 4pm (48.41%)
t2	8am - 4pm (48.41%)	4pm - 12am (37.94%)	12am - 8am (13.64%)
t3	12am - 8am (13.64%)	8am - 4pm (48.41%)	4pm - 12am (37.94%)

Table 6.4: Traffic distribution for each time period of different regions with RS #1

Time Periods	Region 1	Region 2	Region 3	Region 4
t1	4pm - 12am (37.94%)	12am - 8am (13.64%)	8am - 4pm (48.41%)	4pm - 12am (37.94%)
t2	8am - 4pm (37.94%)	4pm - 12am (13.64%)	12am - 8am (48.41%)	8am - 4pm (37.94%)
t3	12am - 8am (37.94%)	8am - 4pm (13.64%)	4pm - 12am (48.41%)	12am - 8am (37.94%)

Table 6.5: Traffic distribution for each time period in each region with RS #2

Several sets of requests are generated with these three different traffic pattern. Each request with different living time periods is randomly assigned to a node, which is the source node of that request, so that the traffic volume based on each node at one time period is random. However, the total traffic volume is considered varies from 50 to 200 units. In order to make the data simpler and the results more comparable, we define the synchronization factor $\delta_v = 0.1$ for all node v. Also, there are only requests survive for at most 2 time periods, and no one survive for all three time periods.

6.1.4 Data Sets

Finally, there are four test data sets generated, and Table 6.6 display the data centers, number of regions and traffic pattern used in each data set.

data sets	data centers	regions	traffic pattern
DS #1	DC #1	RS #1	Pattern #1
DS #2	DC #1	RS #1	Pattern #2
DS #3	DC #2	RS #2	Pattern #2
DS #4	DC #2	RS #2	Pattern #3

Table 6.6: Data centers, number of regions and traffic pattern used in each data set

- *DS #1*: Pattern #1 is used to generate requests for this data set. These requests will be implemented on network which is divided into 3 regions with set of data center DC #1. As shown in Table 6.7, in time period $t1$ of Region 1, 11% is the traffic only appears in single time period, and 3% is the traffic go through two time periods. The two percentage is based on the total traffic volume originating in Region 1.

- *DS #2*: Traffic is generated based on Pattern #2. We use network divided into 3 regions with DC #1. The traffic distribution in every time period in each region are illustrated in Table 6.8. .

- *DS #3*: The network used in this data set is different than the two above. We divided the USA network with 4 regions and use DC #2. Requests are generated by considering Pattern #2. The percentage of traffic distributed to each region in each time period is shown in Table 6.9. .

- *DS #4*: Network with RS #2 and DC #2 is used. Traffic pattern for this data set is Pattern #3. Table 6.9 described the traffic volume for each time period in each region. The number in the table can be read same way as Table 6.10.

	Region 1 (33.33%)	Region 2 (37.5%)	Region 3 (29.17%)
$t1$	11% ($t1$) + 3% ($t1$-$t2$)	30% ($t1$) + 8% ($t1$-$t2$)	39% ($t1$) + 10% ($t1$-$t2$)
$t2$	30% ($t2$) + 8% ($t2$-$t3$)	39% ($t2$) + 10% ($t2$-$t3$)	11% ($t2$) + 3% ($t2$-$t3$)
$t1$	39% ($t3$) + 10% ($t3$-$t1$)	11% ($t3$) + 3% ($t3$-$t1$)	30% ($t3$) + 8% ($t3$-$t1$)

Table 6.7: Traffic distribution in DS #1 (e.g. in time period $t1$ of Region 1, 11% is the traffic only appears in single time period, and 3% is the traffic go through two time periods.)

6.2 Results of Model 4 with Parallel Solution Strategy

In this section, we present the results for the four data sets we used, and do some analysis based on these results. Section 6.2.1 shown the results for DS #1 and DS #2. The results got from DS #3 and DS #4 is displayed in Section 6.2.2.

	Region 1 (33.33%)	Region 2 (37.5%)	Region 3 (29.17%)
$t1$	3% ($t1$) + 11% ($t1$-$t2$)	8% ($t1$) + 30% ($t1$-$t2$)	10% ($t1$) + 39% ($t1$-$t2$)
$t2$	8% ($t2$) + 30% ($t2$-$t3$)	10% ($t2$) + 39% ($t2$-$t3$)	3% ($t2$) + 11% ($t2$-$t3$)
$t3$	10% ($t3$) + 39% ($t3$-$t1$)	3% ($t3$) + 11% ($t3$-$t1$)	8% ($t3$) + 30% ($t3$-$t1$)

Table 6.8: Traffic distribution in DS #2

	Region 1 (29.17%)	Region 2 (16.67%)	Region 3 (25%)	Region 4 (29.17%)
$t1$	3% ($t1$) + 11% ($t1$-$t2$)	8% + 30%	10% + 39%	3% + 11%
$t2$	8% ($t2$) + 30% ($t2$-$t3$)	10% + 39%	3% + 11%	8% + 30%
$t3$	10% ($t3$) + 39% ($t3$-$t1$)	3% + 11%	8% + 30%	10% + 39%

Table 6.9: Traffic distribution in DS #3 (e.g. in time period $t1$ of Region 1, 11% is the traffic only appears in single time period, and 3% is the traffic go through two time periods.)

6.2.1 Bandwidth Requirements with Time-varying Traffic

The relative change in bandwidth cost (i.e., the first summation of the optimization objective (4.78)) for the various scenarios is shown in Fig. 6.4. From these numerical results, we learn that the total bandwidth cost is reduced with average 5.1% (resp. 6.4%) for Scenario II (resp. Scenario III) with traffic Pattern #1, and by 6.9% (resp. 8.2%) with Pattern #2 (where the average is taken over all traffic instances). This net saving mainly stems from a reduction of bandwidth for the backup paths, due to increased sharing: we noted that an average reduction of the backup bandwidth cost with average 11.5% (resp. 13.4%) for Pattern #1 and 14.2% (resp. 16.3%) for Pattern #2, for Scenario II (resp. Scenario III). We verified

	Region 1 (29.17%)	Region 2 (16.67%)	Region 3 (25%)	Region 4 (29.17%)
$t1$	7% ($t1$)+7% ($t1$-$t2$)	19%+19%	24%+24%	7%+7%
$t2$	19% ($t2$)+19% ($t2$-$t3$)	24%+24%	7%+7%	19%+19%
$t3$	24% ($t3$)+24% ($t3$-$t1$)	7%+7%	19%+19%	24%+24%

Table 6.10: Traffic distribution in DS #4

that these savings do not require all 2-period traffic requests to change their routing when going from one period to the next, but only about half of them. Further, compared with the results got from the single-time period model, these preliminary results suggest that the cost advantage can be achieved by only changing the backup/synchronization paths when we consider multiple time periods together (Scenario II): The advantage of allowing also the working path to be changed is much smaller, which is just reversed in the results of single-time period model.

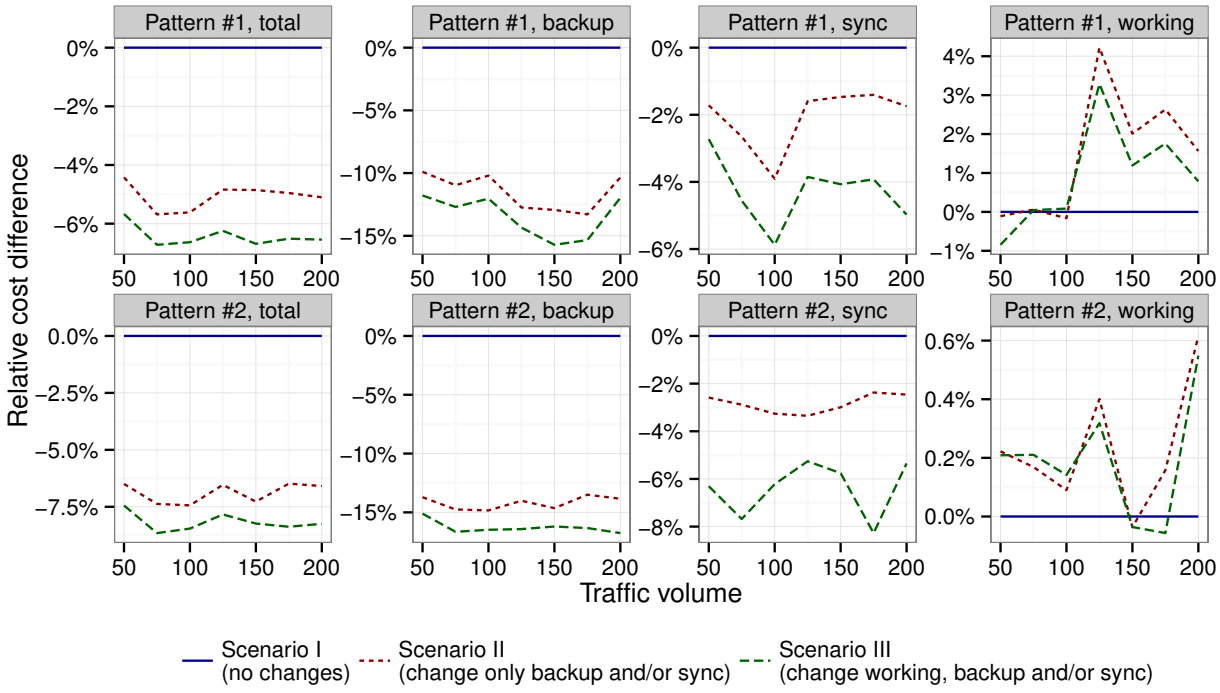

Fig. 6.4: Bandwidth Saving got from DS #1 and DS #2

From the results presented above, the observation is as follows:

- The reduction of bandwidth requirements is mainly depended on the saving of backup paths.
- Compared with Scenario I, the bandwidth requirements on working paths are slightly higher while the overall bandwidth requirements are less. When backup paths have the freedom to change, the less optimal working paths will be chosen in order to achieve better backup sharing.

- When the working paths are allowed to be rerouted (Scenario III), the bandwidth saving is better than the situation that only backup paths can be rerouted (Scenario II). But the difference is not significant.

- Compared with Scenario I, the bandwidth saving is not more than 10% (in Scenario III, even smaller for Scenario II) which is quite small.

6.2.2 More Experiments and Results

Since the results we got from DS #1 and DS #2 show that the saving is no more than 10%, we try some "extreme "cases with data centers which are almost in a line (see Fig. 6.2). The data sets we used are DS #3 and DS #4.

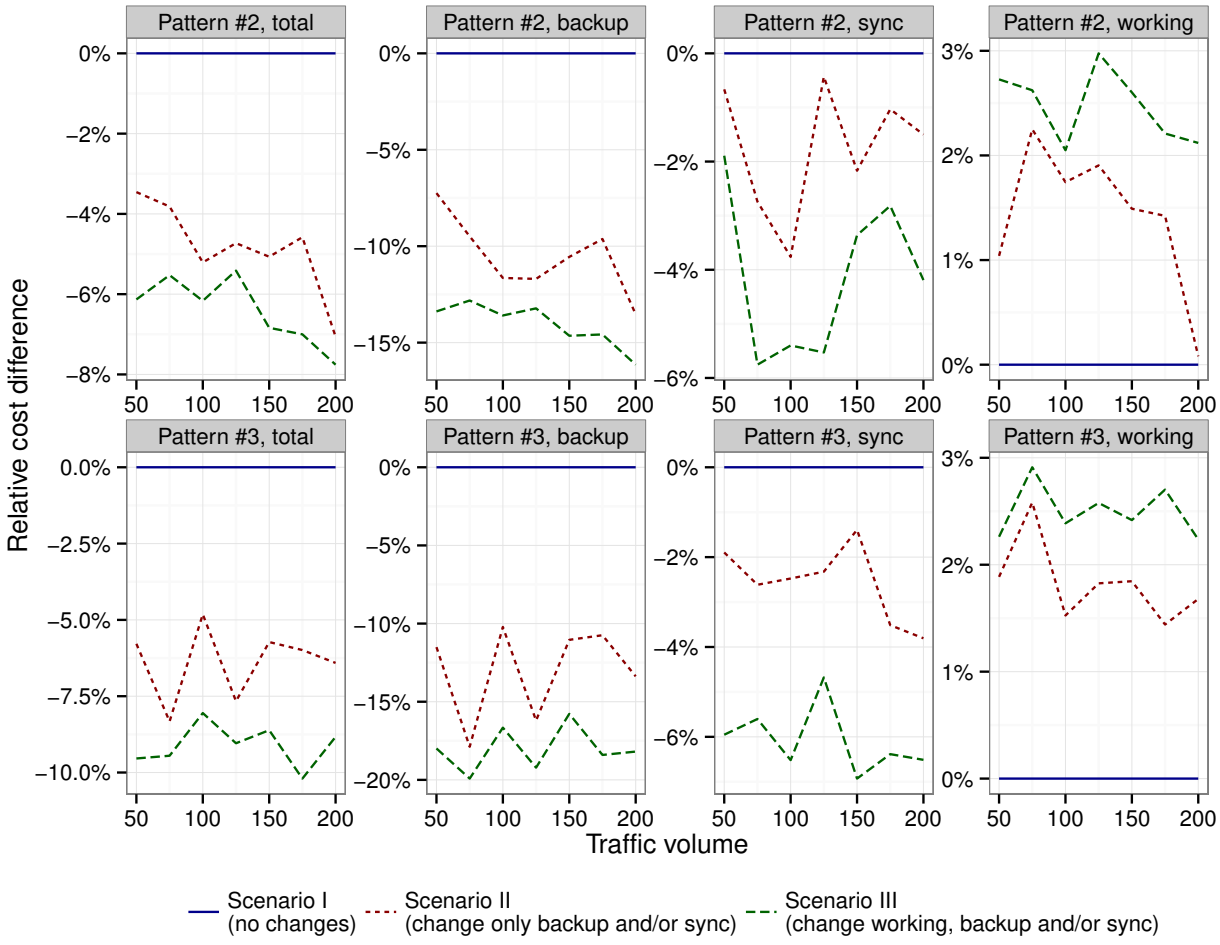

Fig. 6.5: Bandwidth Saving got from DS #3 and DS #4

From these numerical results illustrated in Fig. 6.5, we see the total bandwidth cost is reduced up to 5.6% on average (resp. 6.9%) for Scenario II (resp. Scenario III) with traffic DS #3, and by 6.8% (resp. 9%) with DS #4 (where the average is taken over all traffic instances). The average reduction of the backup bandwidth cost is up to 11.8% (resp. 16.4%) for DS #3 and 15.1% (resp. 18%) for DS #4, for Scenario II (resp. Scenario III).

The results are similar to what we got from DS #1 and DS #2, we didn't see any obvious improvement of bandwidth saving.

6.3 Comparison of Parallel Strategy and Serial Strategy

In order to explain the advantage of using parallel strategy, we compare it with a serial strategy (only solve one pricing problem for a single node each time). Here, if the pricing problems for all nodes in every time period are solved once (PP(v,t), $v \in V, t \in T$), we say that a "round" computation is completed. We compare the number of rounds cost by these two strategy in Section 6.3.1. The CPU time comparison is presented in Section 6.3.2.

6.3.1 The Number of Rounds Cost by Parallel Strategy and Serial Strategy

Fig. 6.6 and Fig. 6.7 are the comparison for experiments based on DS #3. The bandwidth requirements for the first 5 rounds are listed in Table 6.11. The results for other traffic pattern and scenarios are similar.

	0	1	2	3	4	5
Parallel	1,650,000	248,852.9	190,711.5	179,291.75	169,220.63	165,277.88
Serial	1,650,000	240,106	183,014	170,407	164,256	163,102.19

Table 6.11: Bandwidth requirements for the first 5 rounds with DS #3

From the results, we can see that the bandwidth requirements decrease sharply in the first 5 rounds, especially the first round. It is obviously, in one round, serial strategy can

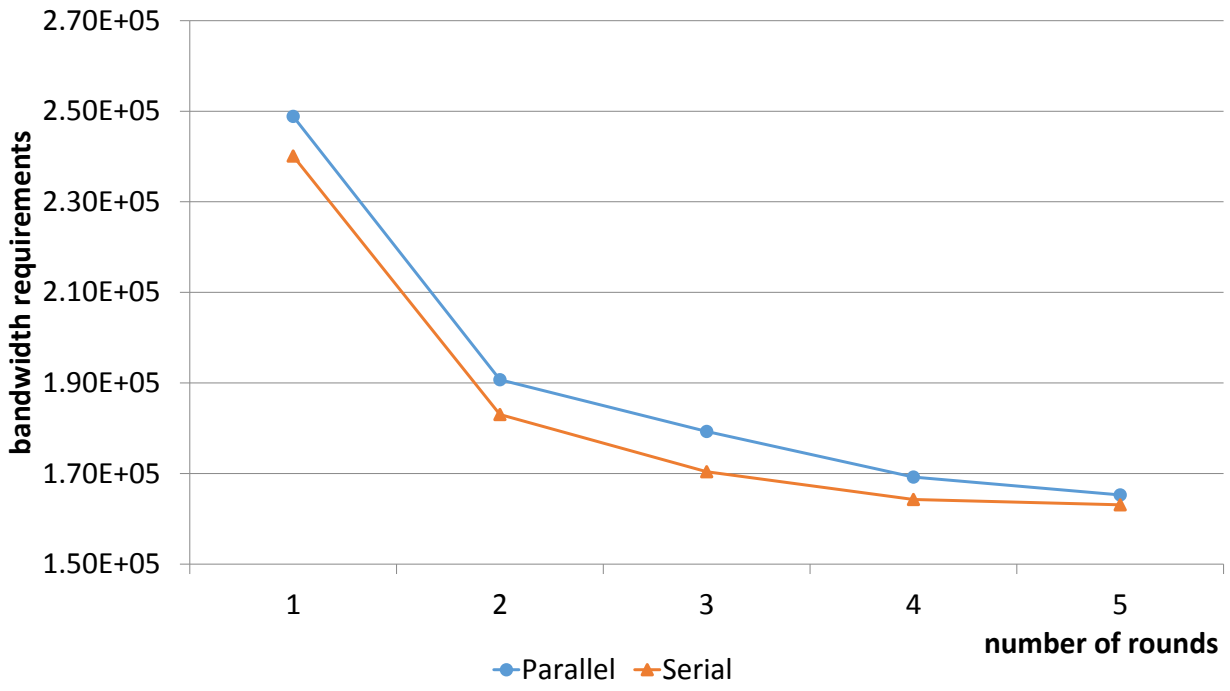

Fig. 6.6: Comparison of Number of Rounds in Parallel Strategy and Serial Strategy for the First 5 Rounds with DS #3

get better improvement with bandwidth requirements. So that, in order to get the optimal solution, parallel strategy need more rounds of computation (from the results, the difference is approximately 10 rounds). However, the pricing problems in a round with serial strategy is solved one by one, and with parallel strategy, all the PPs for different nodes can be solved at same time.

6.3.2 Comparison of CPU Time

We compare the CPU time in parallel strategy and serial strategy with same data set used in Section 6.3.1.

Fig. 6.8 shows the CPU time for each round. In each round, the CPU time for serial strategy is approximately 19 times more than parallel strategy. The overall CPU time consumption after each round of computation is illustrated in Fig. 6.9. After the RVNM problem is completely solved, the overall CPU time cost by serial strategy is about 13.8 times more than the parallel strategy.

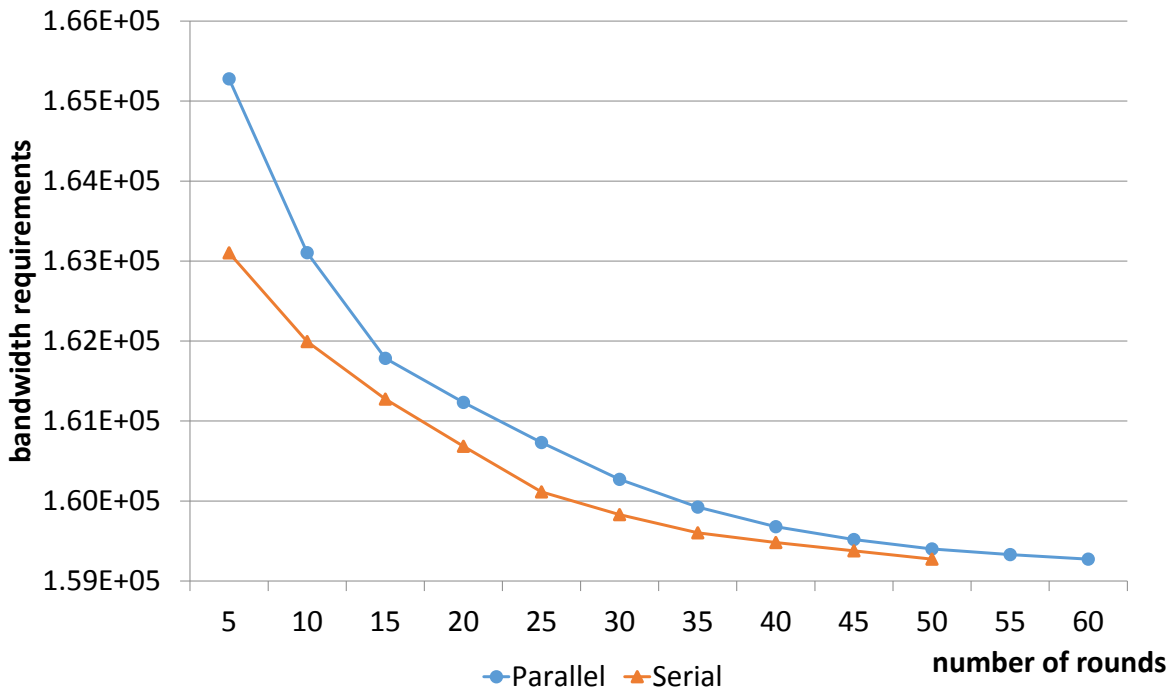

Fig. 6.7: Number of Rounds Comparison of Parallel Strategy and Serial Strategy (After the 5th Round) with DS #3

Most of time is used to solve master problems. There are no columns at the very beginning, therefore master problem can be solved very quickly (do not need to choose any configurations since the configuration set is empty). But after that, for parallel strategy, solving MP need 47 times more CPU time than PP on average. For serial strategy, the CPU time of MP are computed by summing the time consumption for all iterations of MP in a round (same for PP). The CPU time which spend on MP is about 53 times more than PP on average.

We can form these results by using parallel strategy, we need more processors to solve the problem, but we could gain much CPU time saving. The saving is not only due to parallelization of PP computation, also because the reduction of the number of iterations that MP is being solved.

In this chapter, we presented the results that is got from several experiments for our Model 4, and illustrated the comparison of parallel strategy and serial strategy, listed our investigation. The conclusion will be summarized in next chapter.

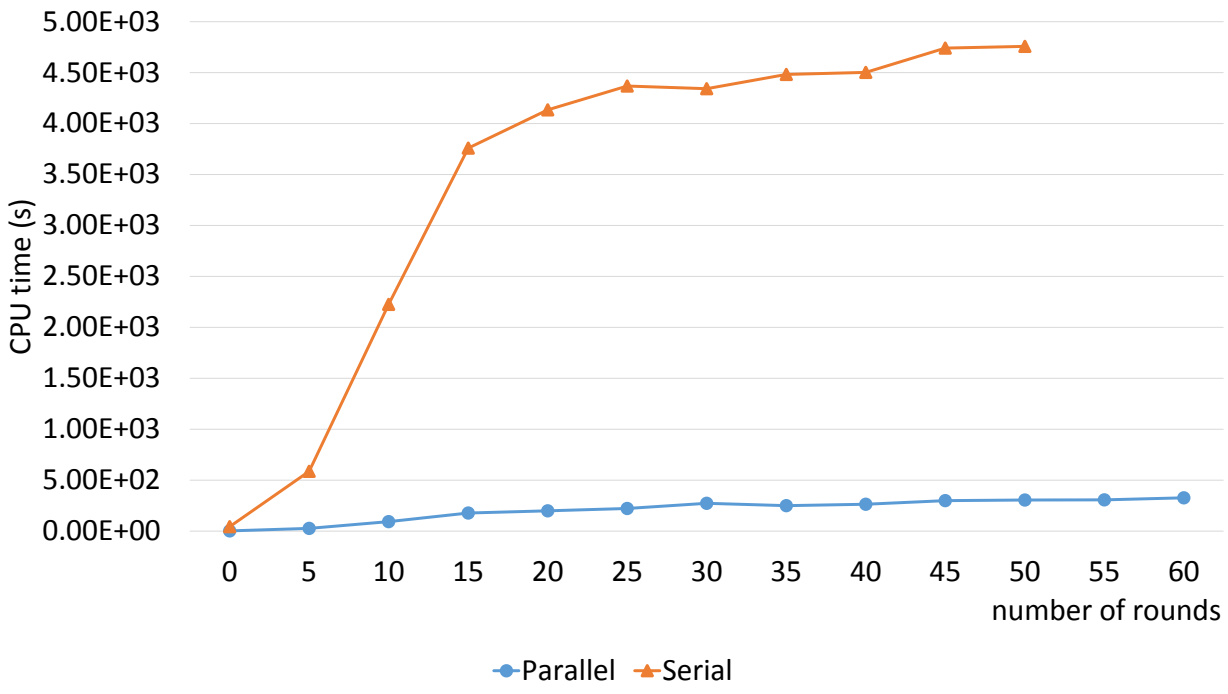

Fig. 6.8: CPU Time Cost by Each Round with DS #3

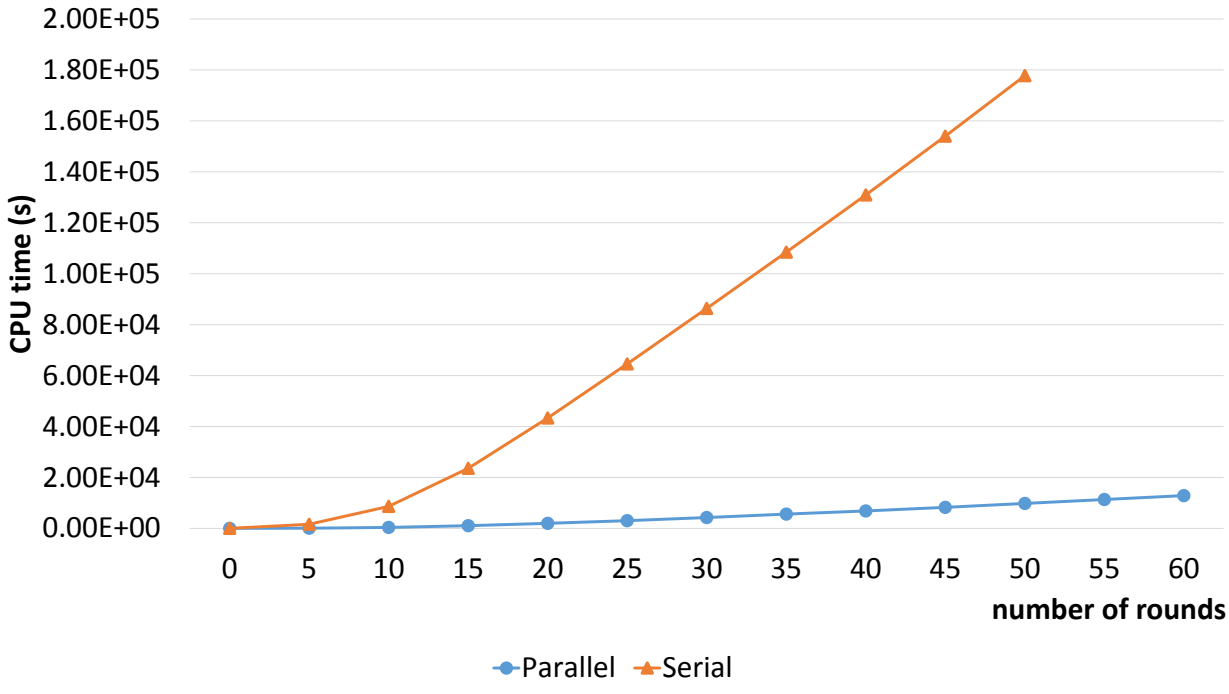

Fig. 6.9: Overall CPU Time with DS #3

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Based on the results we got and the analysis of our model, we got several conclusions given below:

- By using parallel strategy, we need more rounds to get the optimal solution compared with the serial strategy. But we can gain much CPU time saving with the parallel strategy. The overall saving is mainly cased by the reduction of the number of iterations that MP is solved. If we look at the CPU time spent by PP, the results also indicate that by using parallel strategy, solving PP is almost 14 times faster on average.

- By rerouting both backup and synchronization paths we could get more bandwidth saving. However, by using present data instances, we only got around 10% bandwidth saving when compared Scenario III and Scenario I. Eeven a little bit less when compare Scenario I and Scenario II. This indicated that it is not really worth to reroute backup.

- No real need (at least not much from the results we got) to fix the 'non' accuracy of Scenario II. The 'non' accuracy caused by the assumption we made for model 4. If traffic increase 5 units from one time period to next which included 5 units dropped and 10 units new. We force 5 units of the new traffic to use working paths assigned to the dropped traffic, in other words, we consider there are only 5 units of new traffic. Therefore, the results we got for scenario 2 could be slightly higher than the accuracy

results. From the results shown in Chapter 6, we can see that the difference between scenario 2 and 3 is only about 2-3% which is really small, and the accuracy results for scenario 2 will make it even smaller. so it will not influence our conclusion : It is not worth to reroute the working paths, since rerouting is costly and the benefit is limited.

7.2 Future Work

There are some work related to this thesis, we could leave them for further.

- Make an analysis of whether it's worth to implement a model to optimize the number of rerouting (model 3). Model 3 has the ability to distinguish dropped and new traffic, and could lead to a accuracy result. Meanwhile, we could also optimize the number of rerouted traffic. The problem of this model, as mentioned previously, is the linearization of the master model. If we could find a efficient way to solve the linearization problem, it maybe worth to implement this model.

- Better understanding of why we do not get more benefit. The results for Scenario II and Scenario III show that we do not get much saving by rerouting. We could check out the details of the results we got to analyze the reason, and figure out whether results will be significantly effected by traffic pattern and DC location.

- Do experiments with real data. In this thesis, we generate test data by using the traffic pattern recorded by a website. We could try some real data in the future to do further research on this topic.

Bibliography

[1] 10 fool-proof predictions for the internet in 2020. http://www.networkworld.com/article/2238913/wireless/10-fool-proof-predictions-for-the-internet-in-2020.html. Last visit 05/Nov/2015.

[2] Cloud storage architecture. https://commons.wikimedia.org/wiki. Last visit 05/Nov/2015.

[3] internetlivestats.com. http://www.internetlivestats.com. Last visit 08/April/2015.

[4] World internet users statistics and 2015 world population stats. http://internetworldstats.com/stats.htm. Last visit 19/Sept/2015.

[5] M. Baker, B. Carpenter, G. Fox, S. Ko, and S. Lim. mpijava: an object-oriented java interface to mpi. *1st Intl. Workshop on Java for Parallel and Distributed Computing (IWJPDC99), San Juan, Puerto Rico, pp. 748762.*, 1999.

[6] M. Baker, B. Carpenter, and A. Shaf. Mpj express: Towards thread safe java hpc. *Proc. 8th IEEE Intl. Conf. on Cluster Computing (CLUSTER06), Barcelona, Spain, pp. 110.*, 2006.

[7] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[8] S. Bhattacharjee, M. Ammar, E. Zegura, V. Shah, and Z. Fei. Application-layer anycasting. *Proc. INFOCOM*, April 1997.

[9] M. Bui, B. Jaumard, and C. Develder. Anycast end-to-end resilience for cloud services over virtual optical networks. In *15th International Conference on Transparent Optical Networks (ICTON)*, pages 1–7. IEEE, 2013.

[10] M. Bui, B. Jaumard, and C. Develder. Anycast end-to-end resilience for cloud services over virtual optical networks (invited). In *Proc. 15th Int. Conf. Transparent Optical Netw. (ICTON 2013)*, Cartagena, Spain, 23–27 Jun. 2013.

[11] M. Bui, B. Jaumard, and C. Develder. Resilience options for provisioning anycast cloud services with virtual optical networks. In *IEEE International Conference on Communications (ICC)*, pages 3462–3468, June 2014.

[12] B. Carpenter, V. Getov, G. Judd, A. Skjellum, and G. Fox. Mpj: Mpilike message passing for java. *Concurrency: Practice and Experience, vol. 12, no. 11, pp. 10191038*, 2000.

[13] N. Chowdhury, M. Kabir, and R. Boutaba. A survey of network virtualization. *Computer Networks 54.5 (2010): 862-876.*, 18(10):2062–2071.

[14] K. Christodoulopoulos, E. Varvarigos, C. Develder, M. De Leenheer, and B. Dhoedt. Job demand models for optical grid research. pages 127–136, 2007.

[15] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[16] C. Develder, J. Buysse, M. De Leenheer, B. Jaumard, and B. Dhoedt. Resilient network dimensioning for optical grid/clouds using relocation. pages 6262–6267, 2012.

[17] C. Develder, J. Buysse, A. Shaikh, B. Jaumard, M. De Leenheer, and B. Dhoedt. Survivable optical grid dimensioning: anycast routing with server and network failure protection. In *IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.

[18] C. Develder, M. De Leenheer, B. Dhoedt, M. Pickavet, D. Colle, F. De Turck, and P. Demeester. Optical networks for grid and cloud computing applications. *Proceedings of the IEEE*, 100(5):1149–1167, May 2012.

[19] C. Develder, B. Dhoedt, B. Mukherjee, , and P. Demester. On dimensioning optical grids and the impact of scheduling. *Photonic Network Communications*, 17(3):255–265, 2009.

[20] C. Develder, B. Dhoedt, B. Mukherjee, and P. Demeester. On dimensioning optical grids and the impact of scheduling. *Photonic Network Communications*, 17(3):255–265, 2009.

[21] R. Duncan. A survey of parallel computer architectures. *Computer*, 23(2):5–16, Feb 1990.

[22] R. Elmasri. *e-Study Guide for: Operating Systems*. 7 Nov 2014.

[23] M. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, Sept 1972.

[24] I. Foster. What is the grid? a three point checklist. *ThreePoint-Check*, 2006.

[25] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable vritual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, March 2001.

[26] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008.

[27] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Mathematical programming*, 1(1):168–194, 1971.

[28] X. He and G.-S. Poo. Sub-reconfiguration of backup paths based on shared path protection for wdm networks with dynamic traffic pattern. In *The Ninth International Conference on Communications Systems, 2004. ICCS 2004.*, pages 391–395, Sept 2004.

[29] B. Jaumard, M. Bui, B. Mukherjee, and C. S. Vadrevu. Ip restoration vs. optical protection: Which one has the least bandwidth requirements? *Optical Switching and Networking*, 10(3):261 – 273, 2013.

[30] B. Jaumard, A. Hoang, and M. Bui. Using decomposition techniques for the design of survivable logical topologies. In *IEEE 5th International Conference on Advanced Networks and Telecommunication Systems (ANTS)*, pages 1–6, Dec 2011.

[31] B. Jaumard, A. Hoang, and M. Bui. Path vs. cutset approaches for the design of logical survivable topologies. In *IEEE International Conference on Communications (ICC)*, pages 3061–3066, June 2012.

[32] D. Katabi and J. Wroclawski. A framework for scalable global ip-anycast (gia). *ACM SIGCOMM Computer Communication Review*, 30(4):3–15, 2000.

[33] E. Kolodner, S. Tal, D. Kyriazis, D. Naor, M. Allalouf, L. Bonelli, P. Brand, A. Eckert, E. Elmroth, S. Gogouvitis, D. Harnik, F. Hernandez, M. Jaeger, E. Lakew, J. Lopez, M. Lorenz, A. Messina, A. Shulman-Peleg, R. Talyansky, A. Voulodimos, and Y. Wolfsthal. A cloud environment for data-intensive storage services. In *Third IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 357–366, Nov 2011.

[34] H. Li. Identity-based grid authentication protocol, June 6 2012. CN Patent App. CN 201,010,568,925.

[35] C. Liu and L. Ruan. A new survivable mapping problem in ip-over-wdm networks. *IEEE Journal on Selected Areas in Communications*, 25(3):25–34, April 2007.

[36] R. Louwersheimer. Implementing anycast in ipv4 networks.

[37] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[38] D. A. Mallón, G. L. Taboada, J. Tourio, and R. Doallo. Npb-mpj: Nas parallel benchmarks implementation for message-passing in java. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 181–190. IEEE, 2009.

[39] P. Mell and T. Grance. The nist definition of cloud computing. 2011.

[40] C. Metz. Ip anycast point-to-(any) point communication. *Internet Computing, IEEE*, 6(2):94–98, Mar 2002.

[41] E. Modiano and A. Narula-Tam. Survivable routing of logical topologies in wdm networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 348–357 vol.1, 2001.

[42] E. Modiano and A. Narula-Tam. Survivable lightpath routing: a new approach to the design of wdm-based networks. *IEEE Journal on Selected Areas in Communications*, 20(4):800–809, May 2002.

[43] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C. Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Transactions on Networking (ToN)*, 15(2):359–372, 2007.

[44] G. L. Nemhauser. Column generation for linear and integer programming. *Optimization Stories*, 20:64, 2012.

[45] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[46] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. Technical report, 1993.

[47] J. Pourqasem, S. Karimi, and S. Edalatpanah. Comparison of cloud and grid computing. *American Journal of Software Engineering*, 2(1):8–12, 2014.

[48] K. Rapoza. "chinas weibos vs uss twitter: And the winner is?". *Forbes.*, Retrieved 4 August 2011.

[49] D. A. Schupke and R. G. Prinz. Capacity efficiency and restorability of path protection and rerouting in wdm networks subject to dual failures. *Photonic Network Communications*, 8(2):191–207, 2004.

[50] A. Shaikh. *Optical Grid Network Dimensioning, Provisioning, and Job Scheduling*. PhD thesis, Concordia University, 2014.

[51] A. Shaikh, J. Buysse, B. Jaumard, and C. Develder. Anycast routing for survivable optical grids: scalable solution methods and the impact of relocation. *Journal of Optical Communications and Networking*, 3(9):767–779, 2011.

[52] S. Soudan, B. Chen, and P. Primet. Flow scheduling and endpoint rate control in grid networks. *Future Generation Computer Systems*, page 25(8):904911, 2009.

[53] G. L. Taboada, J. Tourino, and R. Doallo. Performance analysis of java message-passing libraries on fast ethernet, myrinet and sci clusters. *Proc. 5th IEEE Intl. Conf. on Cluster Computing (CLUSTER03) pp.118–126*, 2003.

[54] G. L. Taboada, J. Touriño, and R. Doallo. F-mpj: scalable java message-passing communications on parallel systems. *The Journal of Supercomputing*, 60(1):117–140, 2012.

[55] A. Todimala and B. Ramamurthy. A scalable approach for survivable virtual topology routing in optical wdm networks. *IEEE Journal on Selected Areas in Communications*, 25(6):63–69, August 2007.

[56] F. Travostino, J. Mambretti, and G. Karmous-Edwards. *Grid Networks: Enabling Grids with Advanced Communication Technology*. Wiley, 2006.

[57] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage. California fault lines: Understanding the causes and impact of network failures. *SIGCOMM Comput. Commun. Rev.*, 40(4):315–326, Aug. 2010.

[58] J. L. Vizcaíno, Y. Ye, V. López, F. Jiménez, F. Musumeci, M. Tornatore, A. Pattavina, and P. M. Krummrich. Protection in optical transport networks with fixed and flexible grid: Cost and energy efficiency evaluation. *Optical Switching and Networking*, 11:55–71, 2014.

[59] H. Zang, C. Ou, and B. Mukherjee. Path-protection routing and wavelength assignment (rwa) in wdm mesh networks under duct-layer constraints. *IEEE/ACM Transactions on Networking (TON)*, 11(2):248–258, 2003.

[60] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C. Chuah. Failure inferencing based fast rerouting for handling transient link and node failures. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2859–2863. IEEE, 2005.