Efficient Virtual Network Embedding onto A Hierarchical-Based Substrate Network Framework

by

Tay Ghazar

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Electrical and Computer Engineering

Supervisor
Dr. Nancy Samaan

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

The current Internet architecture presents a barrier to accommodate the vigorous arising demand for deploying new network services and applications. The next-generation architecture views the network virtualization as the gateway to overcome this limitation. Network virtualization promises to run efficiently and securely multiple dedicated virtual networks (VNs) over a shared physical infrastructure. Each VN is tailored to host a unique application based on the user's preferences.

This thesis addresses the problem of the efficient embedding of multiple VNs onto a shared substrate network (SN). The contribution of this thesis are twofold: **First**, a novel hierarchical SN management framework is proposed that efficiently selects the optimum VN mapping scheme for the requested VN from more than one proposed VN mapping candidates obtained in parallel. In order to accommodate the arbitrary architecture of the VNs, the proposed scheme divides the VN request into smaller subgraphs, and individually maps them on the SN using a variation of the exact subgraph matching techniques.

Second, the physical resources pricing policy is introduced that is based on *time-of-use*, that reflects the effect of resource congestion introduced by VN users. The preferences of the VN users are first represented through corresponding demand-utility functions that quantify the sensitivity of the applications hosted by the VNs to resource consumption and time-of-use. A novel model of time-varying VNs is presented, where users are allowed to *up- or down-scale* the requested resources to continuously maximize their utility while minimizing the VNs embedding cost.

In contrast to existing solutions, the proposed work does not impose any limitations on the size or topology of the VN requests. Instead, the search is customized according to the VN size and the associated utility. Extensive simulations are then conducted to demonstrate the improvement achieved through the proposed work in terms of network utilization, the ratio of accepted VN requests and the SP profits.

List Of Publications

The following is a list of articles containing various aspects of the work presented in this thesis:

Publications in Refereed Conference Proceedings

Ghazar, T. and Samaan, N., "Hierarchical Approach for Efficient Virtual Network Embedding Based on Exact Subgraph Matching," *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1 – 6, 5 – 9 Dec. 2011

Publications Submitted to Refereed Journals

Ghazar, T. and Samaan, N., "Pricing Utility-Based Virtual Networks for Efficient Embedding onto a Shared Substrate Network", IEEE Transactions on Network and Service Management, (second round of revisions), TNSM-2012-00304

Acknowledgements

At the end of my thesis journey, I would like to thank all those who made this thesis possible.

First of all I would like to express my sincere gratitude to Dr. Nancy Samaan, my research supervisor, for her support, patient guidance and encouragement throughout the course of my studies. I will always be grateful.

Essential support has been always found in my beloved husband for his love and continuous encouragement during the darkest times. I would like to thank my beautiful children, Joy and Jonathan, without them this research would not have even started. I also thank my parents for believing in me, for their love and prayers.

I would like to give my special thanks to my friends Mary, Kirollos and Shenouda whose love enabled me to complete this work.

Above all, I owe it all to Almighty God who made all things possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 An Overview

The Internet has successfully changed the way the world is now communicating. It plays a stunning role in our daily life whether at work or leisure. There has been a continuous enormous growth in both the number of the Internet's users and the number of supported Internet applications. The need for the Internet infrastructure growth and flexibility in adapting new technology is essential for the next generation Internet. However, the Internet is shared among multi-providers that made adopting a new architecture or modifying the existing one extremely difficult. This problem is often referred to the Internet ossification problem [11, 62, 66], and has created a substantial barrier against the future growth of new architecture and applications.

Network virtualization has been introduced as the gateway for the future next-generation architecture. Network virtualization allows multiple virtual networks (VNs) with diversified services and applications to share the underlying resources and simultaneously co-exist. Users of each of the VNs would have the illusion of utilizing a fully dedicated network, possibly, with full administrative control over its allocated resources during its lifetime where new protocols and applications can be deployed. At the same time, virtualization ensures complete isolation and transparency among the co-existing VNs. This isolation achieves the desired security, management and scalability while relieving the infrastructure provider from the detailed management responsibility [23]. With

these properties in mind, network virtualization has become the key enabling technology for creating services in cloud-computing environments. In these environments, VNs are created to host various services that require parallel data processing [60].

1.2 Virtual Network Mapping Challenge

The most challenging problem faced by network virtualization is the efficient mapping of the VNs onto the shared underlying physical network. The processing of mapping of the VNs onto the substrate network is known as the *VN embedding/mapping/matching/assignment problem*. A VN consists of a set of virtual nodes (i.e. routers) that are connected by a set of virtual links with a tailored service or application based network topology. From the SP point of view, the VN embedding problem involves the efficient mapping of the VN's nodes and their corresponding links onto the substrate network that will lead to higher accepted VN requests and maximizes its revenues. The VN embedding problem can be reduced to the multi-way separator problem, which is *NP*-hard [1].

The VN embedding problem has been the focus of many research groups to find the optimum VN mapping algorithm. In order to find a solution, several efficient heuristics have been proposed by restricting the problem in different dimension. A details of some of the proposed heuristics is discussed in Chapter 3.

1.3 Contributions

In this thesis, our contributions can be summarized in the following points:

- We propose a novel hierarchical VN embedding scheme [25] based on a variation of exact subgraph matching techniques [5] tailored for the VN network topology. The scheme performs concurrent searches for more than one efficient VN embedding solution from which a *good* solution is selected. In contrast to existing approaches, scalability and load-balancing are achieved by partitioning the underlying substrate network into a number of virtualization management domains that vary in size at each level of the hierarchy, thus allowing for efficient embedding of VN requests of various sizes. The scheme also efficiently overcomes the SN fragmentation phenomenon

through the periodic repartition of the virtualized sub-domains to better utilize the underlying resources.

- We present a new time-of-use pricing mechanism for VNs, and search for more resource-efficient embedding of the actively dynamic VNs due to the SN congestion [24]. To model price and resource usage sensitivity of the VN requests, we present a new dynamic VN model that employs utility functions to quantify the users' value of the VN resource demands within one cycle.
- We introduce the novel concept of *VN scaling*, where the VN's allocated resources are scaled up or down to maximize the VN owners' utilities and minimize the embedding costs.

1.4 Organization of the Thesis

The remainder of the thesis is organized as follows:

- Chapter 2 provides an overview of the network virtualization architecture, its characteristics and some of the previously proposed projects.
- Chapter 3 discusses the VN embedding problem. A literature review of relevant approaches is also provided.
- Chapter 4 discusses in details our proposed hierarchical substrate management framework and the proposed algorithm for the VN embedding problem.
- Chapter 5 discusses the limitations of the fixed VN dedicated resources and presents a new pricing mechanism for the dynamic utility-based VN embedding scheme based on time-of-use.
- Chapter 6 presents our conclusion and an outline for future research directions.

Chapter 2

Network Virtualization

2.1 Introduction

Over the past decades, the Internet has proved its stunning success and has changed the way the technology is viewed. Massive Internet applications have been introduced to people's daily behavior which has changed their approach towards work, entertainment, shopping, and etc. Unfortunately, the current physical infrastructure is not equipped for this massive and rapid change in the Internet technology. Developers and researchers have shown high interest in adopting a new architectures, however, the infrastructure is shared between many stake holders that impose several restrictions against adopting or performing any physical modifications or upgrades on the existing physical infrastructure [2].

Network virtualization is foreseen as the future gateway that will overcome the current Internet ossification problem [11,62,66] without performing modifications on the physical infrastructure. Network virtualization allows multiple heterogeneous virtual networks (VNs) of variable applications and services to efficiently share the underlying physical resources simultaneously.

In this thesis, we study how network virtualization can facilitate the deployment and management of the emerging Internet applications without modifying the underlying physical architecture. Our work mainly focuses on how to embed the VNs on the substrate network (SN) such that it increase the number of accepted VN requests and

maximizes the substrate network provider's profit.

2.2 Chapter Organization

This chapter is organized as follows:

- Section 2.3 presents a brief historical overview of the network virtualization.
- The components of the network virtualization in presented in Section 2.4.
- The network virtualization business model and their interconnections are discussed in details in Section 2.5.
- The general scheme for the objectives of the network virtualization architecture is presented in Section 2.6.
- Some of the previously presented network virtualization projects are briefly discussed in 2.7.
- Finally, the Virtual Network is defined in Section 2.8.

2.3 Network Virtualization History in a Nutshell

Although virtualization as a concept is not new, however, in the last decade network virtualization has been the focus of many researchers. Network virtualization objective is to introduce a high-level of abstraction that decouples the hardware layer from the virtual layer residing on top of it. Virtual Local Area Network (VLAN), Virtual Private Network (VPN) and overlay network are examples of network virtualization that have been around for quite a few years, each with different services, but with the same goal of abstraction.

2.3.1 Virtual Local Area Networks (VLANs)

A VLAN is a common example of link virtualization. A VLAN represents an isolated network where all its nodes belong to one broadcast domain. However, managing VLANs have shown to be a challenging task [63].

2.3.2 Virtual Private Networks (VPNs)

A VPN is an example of the use of network virtualization where a dedicated network is instantiated upon demand between one or more customer edge (CE) devices that are connected to one or more provider edge (PE) routers [56] that communicate through secure tunnels over the shared underlying physical network. However, VPNs have their limitations [7]:

1. All created VPNs are based on the same network technology.
2. There is no clear separation between the Infrastructure Provider (InP) and VPNs, and they are often offered by the same entity.

2.3.3 Programmable Network Research

The separation of hardware components from that of the software one is the main objective of programmable networks. Programmable network offers interfaces for the construction of network architecture [6].

2.3.4 Overlay Networks

Overlay Networks represent another example of network virtualization that had gained popularity in both experimental and deployment paths. Unlike VPNs, overlay networks have no geographic limitations. Overlays have arbitrary network topologies that depend on the hosted services. Overlays deployment does not require any changes in the underlying infrastructure, thus, they are considered inexpensive, flexible and adaptive to the underlying current network conditions. Overlays have addressed numerous issues with

the current Internet such as end-to-end QoS guarantee [14], protection from Denial-of-Service (DoS) attacks [58], enabling multicasting [3],fault-tolerant [47], content-delivery (CDN) and file sharing [39].

However, overlay network have their limitations [2]:

1. Overlays require high maintenance that made the overlays more suitable for the deployment path than the experimental path.
2. Overlays are mostly created to act as a specific fixes in the network but not as a new Internet architecture in its purist.
3. Most overlays are assumed to be built on top of the IP layer, and thus not open for any new underlying architecture.

2.4 Network Virtualization

Network virtualization allows secured, dedicated and independent network architectures to co-exist and share the underlying physical resources. Network virtualization decouples the services and the applications from the underlying substrate network. There are two main components in network virtualization; link virtualization and node virtualization.

2.4.1 Link Virtualization

In link virtualization, multiple virtual links of different network applications and services are transported over a shared physical link. A virtual link can be mapped on a single substrate link or a simple substrate path (i.e. loopless path) that consists of one or more substrate links.

2.4.2 Node Virtualization

In node virtualization, each single substrate node partitions its physical resources to multiple slices and offers it for lease. A single substrate node can host multiple virtual

nodes that belong to different VNs. The substrate node cannot host more than one virtual node that belongs to the same application. The amount of physical resources dedicated to each VN is based on the application requirements. In addition, a substrate node offers a dedicated and secure traffic transport. Fig. 2.1 shows an example of network virtualization [7], that consists of three nodes (i.e. a, b & c) and two links (i.e. ab and bc). Nodes *a*, *b* & *c* host multiple virtual nodes. Node *b* hosts virtual nodes (i.e. B1 & B2) and forwards traffic from the substrate node *a* to the substrate node *c*, thus, creating a path between the two substrate nodes.

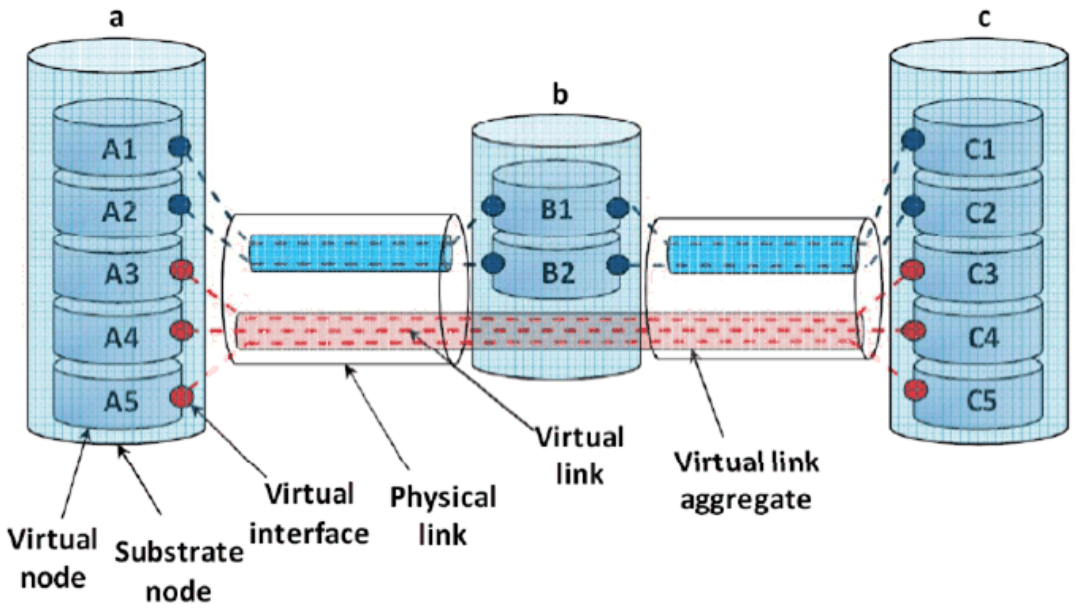

Figure 2.1: Basic Elements of Network Virtualization [7].

2.5 Network Virtualization Business Model

Network virtualization has successfully decoupled the role of the traditional Internet Service Providers (ISPs) into two parts: the InP and the service providers (SPs). The network virtualization model successfully defines three players,

- **The infrastructure providers (InPs)**: The infrastructure providers lease and manage the underlying physical infrastructure. The lessee could be a service provider or

a single user. The InP partitions its resources into isolated and dedicated slices using virtualization techniques. The InP is responsible for accurately describing and advertising its available resources for lease to its users' (e.g. the service providers or other InPs).

There are two types of attributes that describes the underlying infrastructure: Functional attributes and non-functional attributes. Functional attributes represent the network's static parameters like node and link type, geographic location, etc... Non-Functional attributes represent the real-time attributes like the available node capacity (i.e. cpu) and the link capacity (i.e. bandwidth).

The InPs are in charge of forming business relationship with one another, to assure end-to-end service agreement to their customers. The objective of the InP is to efficiently utilize its underlying resources to reduce its costs and to increase its revenues.

- **The virtual networks (VNs)/End-Users**: The end-users represent a variety of network applications and services that simultaneously share the underlying physical resources. The end-users can be VNs or other SPs interested in leasing the physical resources. End users are totally isolated from one another. A VN is mapped on top of one InP or is spanned between multiple InPs. However, each VN can only be managed by one SP. Each SP creates and maintains its own VNs. Once the VN has expired, the SP releases the dedicated resources and makes it available again for future lease.

  Each VN is composed of a set of virtual (logical) nodes that are connected via virtual (logical) links. VNs do not have a strict topology (i.e. VNs can have any arbitrary topology). Each virtual node is mapped on only one substrate node, and each virtual link is mapped on a simple (loopless) substrate path (i.e. one or more substrate link).

- **The service providers (SPs)**: The service provider (i.e. VN provider) acts as the middleware between the InPs that are offering their resources to be leased and the leasers (i.e. end-users) that are interested to lease the network resources for their specific applications or services. The leaser could be another SP or a VN. Based on the end-user constraints, the SP uses an efficient VN mapping algorithm to find a set of available substrate resources and leases them to the end-user. A single SP may collaborate with one or more InPs to create a single VN. Scalability is one of the SP's objectives, as it maximizes its revenues.

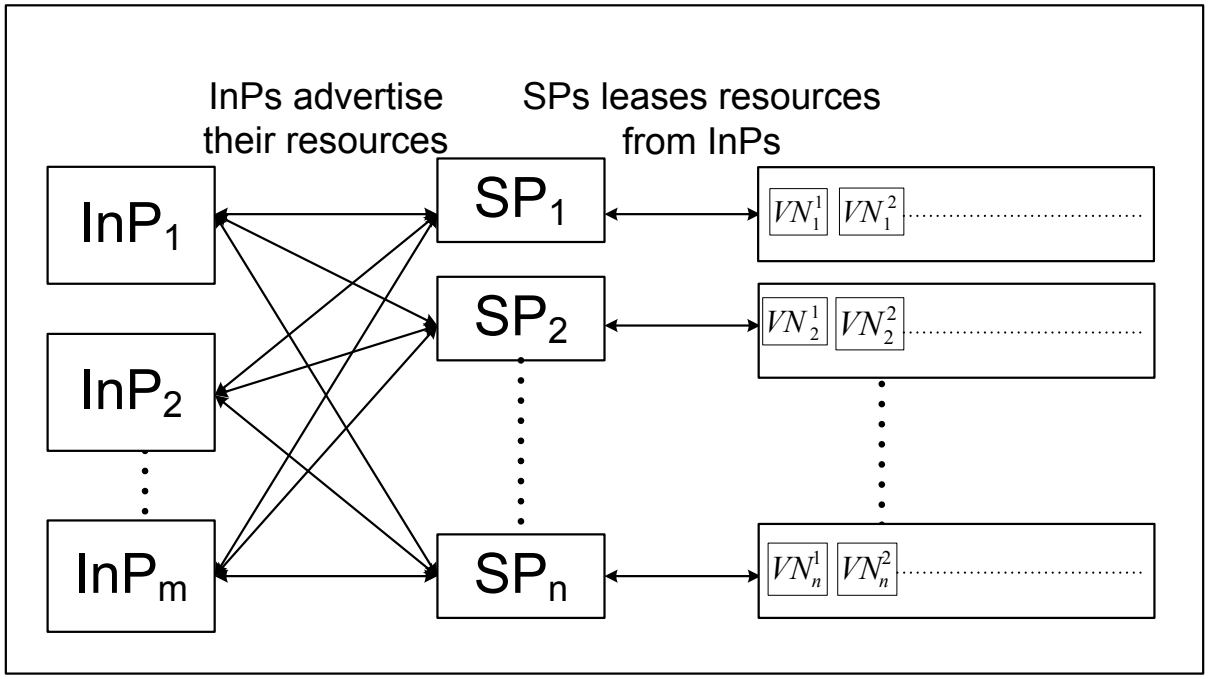

Figure 2.2: network virtualization environment.

Fig. 2.2 shows the interaction between the three players in a network virtualization environment. The InPs and the SPs have many to many relationship. The InPs advertise their available resources for lease. Multiple InPs (i.e. lessees) lease their resources to one or more SPs. The SPs (i.e leasers) might lease resources from one or more InPs to map one single VN.

2.6 Network Virtualization Architectural Objectives

Network virtualization as mentioned above is considered a potential solution that is expected to solve the Internet ossification problem. The idea of having multiple coexisting *virtual networks* on a shared *physical infrastructure* have attracted several research groups and projects. The architecture of the network virtualization involves many players, each with its own financial interest. However, in order for the network virtualization to be beneficial for all the players, the following challenges should be taken into consideration:

- **Flexibility**
  Network virtualization must offer flexibility to all its three players (i.e. InPs, SPs and VNs). The InPs can freely advertise their available resources for lease, and the SPs must have all the flexibility to select from any InP(s) the resources to embed the requested VNs. The SPs also need to be flexible in accommodating any arbitrary VN topology with their specific constraints.

- **Scalability**
  Scalability is one of the main challenges of any SP. Scalability deals with the number of VNs the SP can support and manage. The SPs objective is to lease the maximum amount of their network resources to increase their revenues. However, scalability requires efficient use of the underlying physical resources.

- **Manageability**
  Manageability is very critical for network virtualization. A successful management of network virtualization involves two aspects: First, the InP must have a global view of its underlying technology and is in charge of the efficient management and maintenance of its resources. Each InP is also responsible for the dynamic updates of its current available resources and forwarding the updates to its SPs for future lease.

  Second, a successful coordination between both the InPs and the SPs is critical, SPs as mentioned act as the middle-ware between the InPs and VNs. For the SP to maximize its revenues, it needs to efficiently utilize the InP resources. The SPs manage the VNs by mapping and maintaining them until they expire.

- **Isolation and Security**
  Network virtualization is based on the idea of having multiple VNs that are concurrently sharing the same resources. Thus, isolation and security together are very crucial to ensure privacy and in creation network environment that is resilient to faults and attacks. A security attack or failure in one VN should not affect the other VNs that share the same resources.

- **Accountability**
  Accountability is defined as the Service Level Agreement (SLA) promised by the InPs to their SPs [34]. Accountability also refers to finding a mechanism for failure detection and violation. However, the concept of sharing resources between different SPs makes it very difficult.

- **Cost and QoS**
  InPs must efficiently utilize their network resources to be able to accept more VN requests and increase their revenues. The SPs on the other side must search for the optimum VN embedding scheme that will minimize its cost while guaranteeing the same level of the requested QoS by the VN users.

- **Robustness**
  The InP should guarantee a continue of operation even within any node or link failure.

- **Interoperability**
  Interoperability is an essential issue, as some VNs need to be spanned between multiple InPs. The SPs must guarantee a seamless interoperability across different InPs.

- **Network Technology**
  One of the main objectives of network virtualization that attracted many research groups is its indifference of the underlying network technology.

2.7 Network Virtualization Projects

A number of existing projects has been presented for the network virtualization architecture. The proposed projects were defined as either representing the purist form of architecture or the pluralist form [2]. These projects reflect the path of the Future Internet architecture. The main objective of these projects is to keep the underlying physical infrastructure intact. Some of the proposed projects include:

- **PlanetLab** [48]
  PlanetLab is an overlay-based testbed that is used by the research community to evaluate and deploy geographically distributed network services. It allows multiple isolated services with different topologies to simultaneously run over a shared substrate network. PlanetLab is implemented on top of traditional Internet connectivity (i.e. IP).

- **GENI** [55]
  GENI is an extension to PlanetLab, however, GENI incorporates different types of network architectures like sensor, wireless and mobile networks [35]. GENI does not impose any specific underlying network technology.

- **CABO** [21]
  This project was first proposed as a method to decouple the (InPs) from the SPs, however, coordination between the two providers is a must to guarantee end-to-end network service. Before CABO, network virtualization was perceived as an evaluation platform, however CABO has seen network virtualization capable of supporting multiple simultaneous network architectures.

- **CABERNET** [67]
  CABERNET is an extension to CABO. In this project, the authors introduced the

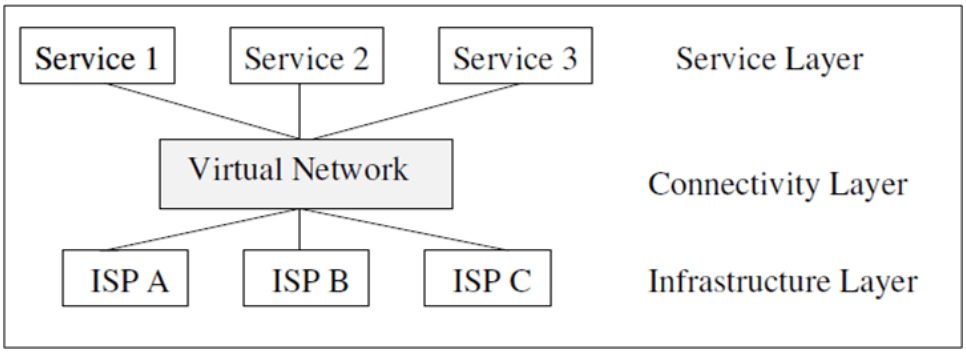

Figure 2.3: CABERNET, a three Layer Network Architecture [67].

connectivity layer. The connectivity layer is a middle layer between the infrastructure providers and the service providers, as shown in Fig. 2.3. The architecture of the connectivity layer represents a one wide virtual network that spans over multiple infrastructure providers. This abstraction layer alleviated the negotiation phase of the service provider with each infrastructure provider. The connectivity layer negotiates prices and offers a better utilization of the underlying physical resources.

- **VNET - 4WARD Project** [51]
  To further separate the infrastructure provider from the service provider, VNET has identified four players for the architecture of the network virtualization as shown in Fig 2.4: The *Physical Infrastructure Provider (PIP)* that owns and manages the physical infrastructure. The *Virtual Network Provider (VNP)* that aggregate the PIPs into one virtual network. The *Virtual Network Operators (VNO)* acts as a connectivity layer between the VNP and the SP. VNO is responsible for mapping the virtual networks onto the VNP. The *Service Provider (SP)* offers its application and services through the virtual network.

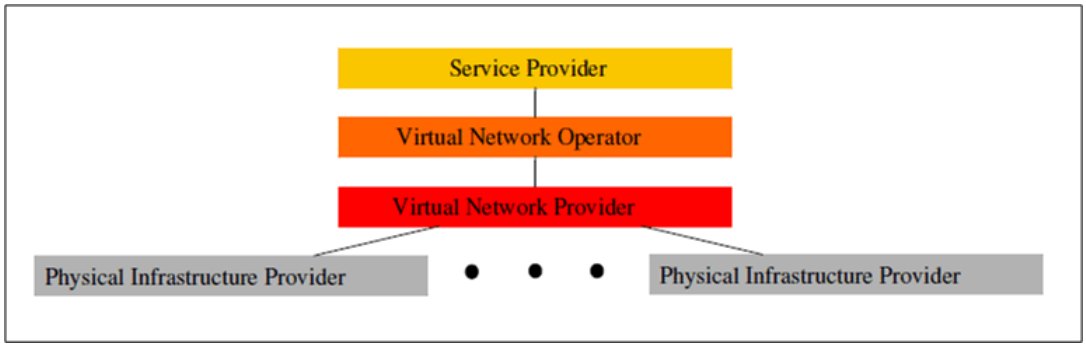

Figure 2.4: VNET Network Architecture [51].

2.8 Virtual Network Embedding Overview

Based on the specific constraints and demands imposed by each VN, the SP *allocates* the required resources and leases them to their end-users (i.e. VNs). This process is called the **VN embedding/mapping**. The VN embedding involves allocating and reserving the requested physical resources to create a VN. The allocated resources is dedicated and isolated from the rest of the substrate network until the VN expires. Once the VN expires, the dedicated resources are released back to the substrate network. The SPs manage the heterogeneous VNs (i.e. end-users) and offer end-to-end services.

Chapter 3

Literature Review

3.1 Introduction

As mentioned in Chapter 2, the main obstacle facing the Internet architectural development is that it is shared between multiple stake-holders, that makes reaching an agreement between them difficult. Network virtualization services solve this problem by introducing the heterogeneous multiple VNs that co-exist simultaneously and share the underlying physical infrastructure. Each VN would have the illusion of utilizing a fully dedicated network, with full administrative control over its allocated resources throughout its lifetime. At the same time, virtualization ensures complete isolation and transparency among the co-existing VNs. This isolation achieves the desired security and management scalability while relieving the infrastructure provider from the detailed management responsibility [23]. Each VN offers a specific service or application based on the end-user's need.

3.2 Chapter Organization

This chapter is organized as follows:

- In Section 3.3, we explore the basic two components of our network model, specifically the substrate network and the virtual network, respectively.

- The main factors and challenges to be considered while designing the VN embedding algorithm are presented in Section 3.4.
- Section 3.5 presents a review of the existing proposed work addressing the VN embedding problem.

3.3 The Network Model

There are two main components in the network model: the substrate network and the virtual network.

3.3.1 The substrate network (SN)

The SN, i.e. the physical infrastructure, is an arbitrary network topology that consists of a set of substrate nodes connected via a set of substrate links. Each substrate node is associated with a specific capacity cpu and geographic location. Each substrate link is associated with the bandwidth capacity. The infrastructure provider (InP) owns and manages the network, and its objective is to efficiently utilize its resource to increase its revenues. The InP advertises its available physical resources to be leased by the service providers (SPs) as discussed in Chapter 2.

3.3.2 The Virtual Network (VN)

The VN is a logical network that may have any arbitrary topology. A virtual network consists of a set of virtual nodes that are connected via a set of virtual links. Each VN is customized to serve a specific network application or a service, and thus imposes specific constraints to be met throughout its lifetime (e.g. cpu and bandwidth constraints, geographic location, maximum delay, minimum throughput, etc.)

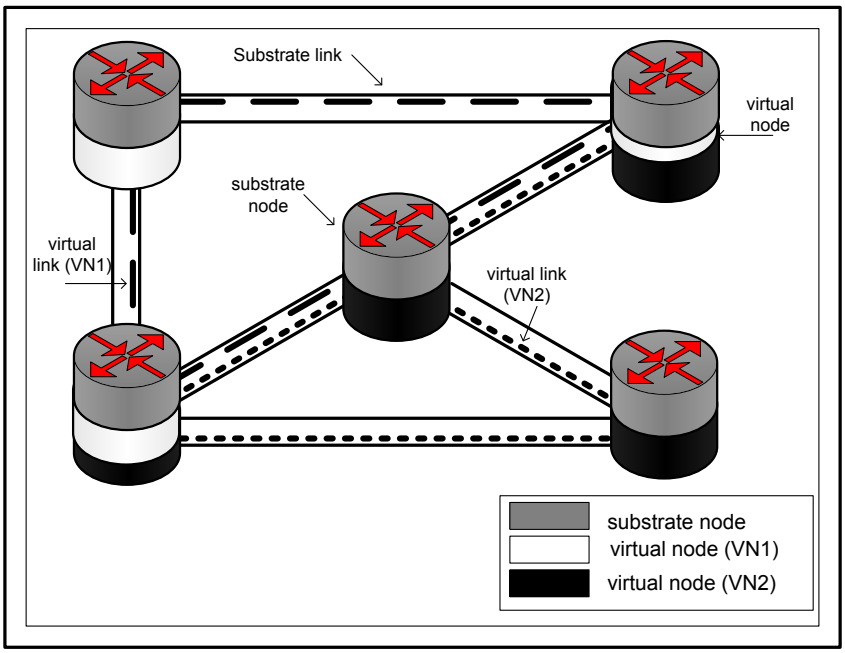

Figure 3.1: Mapped VNs on a substrate network .

3.3.3 The VN embedding Framework

The SN simultaneously hosts multiple VNs that share the underlying resources. The SN offers dedicated and secured resources throughout the lifetime of each VN. Each virtual node in VN is mapped onto a single substrate node on the SN, and each virtual link is mapped onto a simple path between the two mapped substrate nodes. A simple path consists of one or more substrate links. Fig. 3.1 shows a simple example of a SN with 5 routers and 6 substrate links. Two VNs (VN1 and VN2) with different resource constraints are hosted on top of the SN and are sharing the underlying resources.

3.4 The VN embedding problem

As mentioned above, the creation of VNs on top of the underlying physical network(s) is called embedding/mapping. Mapping a VN on top of the substrate network involves mapping all the virtual nodes on the substrate nodes, and all the virtual links on the substrate paths. Each VN specifies its constraints based on its application and expect to have these resources dedicated to it throughout its lifetime.

The VN constraints may include specific geographic constraints, cpu, bandwidth, network topology, throughput, sensitivity to packet delay or loss. When a VN request is received by the SP, the SP performs admission control to determine whether there are sufficient resources to map the request or it should simply reject it. If the SP accepts a VN request, it allocates and dedicates the required resources until the VN expires.

The objective of this thesis is to find an online efficient VN mapping algorithm that will satisfy both the SP and the VN. From the VN perspective an efficient mapping algorithm allows the VN request to be accepted and successfully mapped on top of the substrate network with minimum cost. From the SP perspective, the efficient mapping algorithm is the one that will maximize its revenues. For the SP to maximize its revenues the VN mapping algorithm needs to efficiently utilize its underlying resources to lease the maximum amount of resources while minimizing its VN embedding cost. The SP must provide a load balancing technique so that the underlying resources are not under-utilized. Efficient resource utilization allows the SP increase the number of accepted VN requests and thus increases its revenue.

However, even if all the VN requests are known in advance (i.e. offline approach), still the VN embedding problem can be shown to be NP-hard [1]. The VN embedding problem has attracted several researchers who have proposed different heuristics algorithms to solve the embedding problem. In the next section, a detailed discussion of some of the existing proposed solutions for the VN mapping problem are presented.

3.5 Existing Approaches

3.5.1 Resource Discovery and Management

Once the SP dequeues a VN request, the SP performs admission control to check whether the specific request can be successfully mapped, otherwise it would be either delayed or rejected. The admission control is performed based on the resource discovery scheme used by the SP. The SP needs to continuously perform resource discovery as resources get leased and released back dynamically. Optimum resource discovery leads to maximizing SPs revenues, as the available underlying resources get efficiently utilized. Advertising the available underlying resources falls solely on the InP.

There are two types of attributes that describes the underlying infrastructure: Functional attributes and non-functional attributes. Functional attributes represents the network's static parameters like node and link type, geographic location, etc. Non-Functional attribute represents the real-time attributes like available node capacity (i.e. cpu) and link capacity (i.e. bandwidth), etc.

Fig. 3.2 illustrates an overview of the VN embedding process [29]. In *(step 1)*, the InPs advertise its available resources for lease to its SPs. The advertised resources are sent to a Resource Discovery Framework *(Step 2)*. When the SP receives a VN request as shown in *Step 3*, it tries to match the requested resources with its available resources using a specific mechanism as shown in *(Steps 5-8)* through a resource allocation theme, that will be discussed in Section 3.5.2. The embedding algorithm finds the optimum VN embedding and maps the VN request.

3.5.2 Resource Allocation Themes

In this section, we present and discuss an overview of some of the previous work done in allocating resources for the arrived VN requests and reserving these requests throughout the VNs lifetime. This whole process is also known as the VN mapping/embedding/assignment problem. The problem of the VN assignment can be also reduced to the known multiway separator problem [1], which is NP-hard even if all the VNs are known in advance (i.e. offline). Efficient resource allocation is important in avoiding unnecessary congestion on the SN.

Previous proposals used different techniques to solve the VN embedding problem. In their work, they relied on specific assumption(s) to help them find a solution such as: The offline vs. the online version, specific VN topology vs. arbitrary VN topology, infinite underlying resources vs. limited resources, two stages vs. single stage mapping, and finally static vs. dynamic mapping.

- **The Online vs. The Offline Approach**
  In the online problem [11, 37, 62], the VN requests arrive dynamically to the SP with unpredictable demand or sequence (i.e. arbitrary VN topology, cpu and bw). In the offline problem [9, 38, 53, 66], all the future VN requests and their demands are known in advance. The offline approach makes the VN embedding more optimal compared

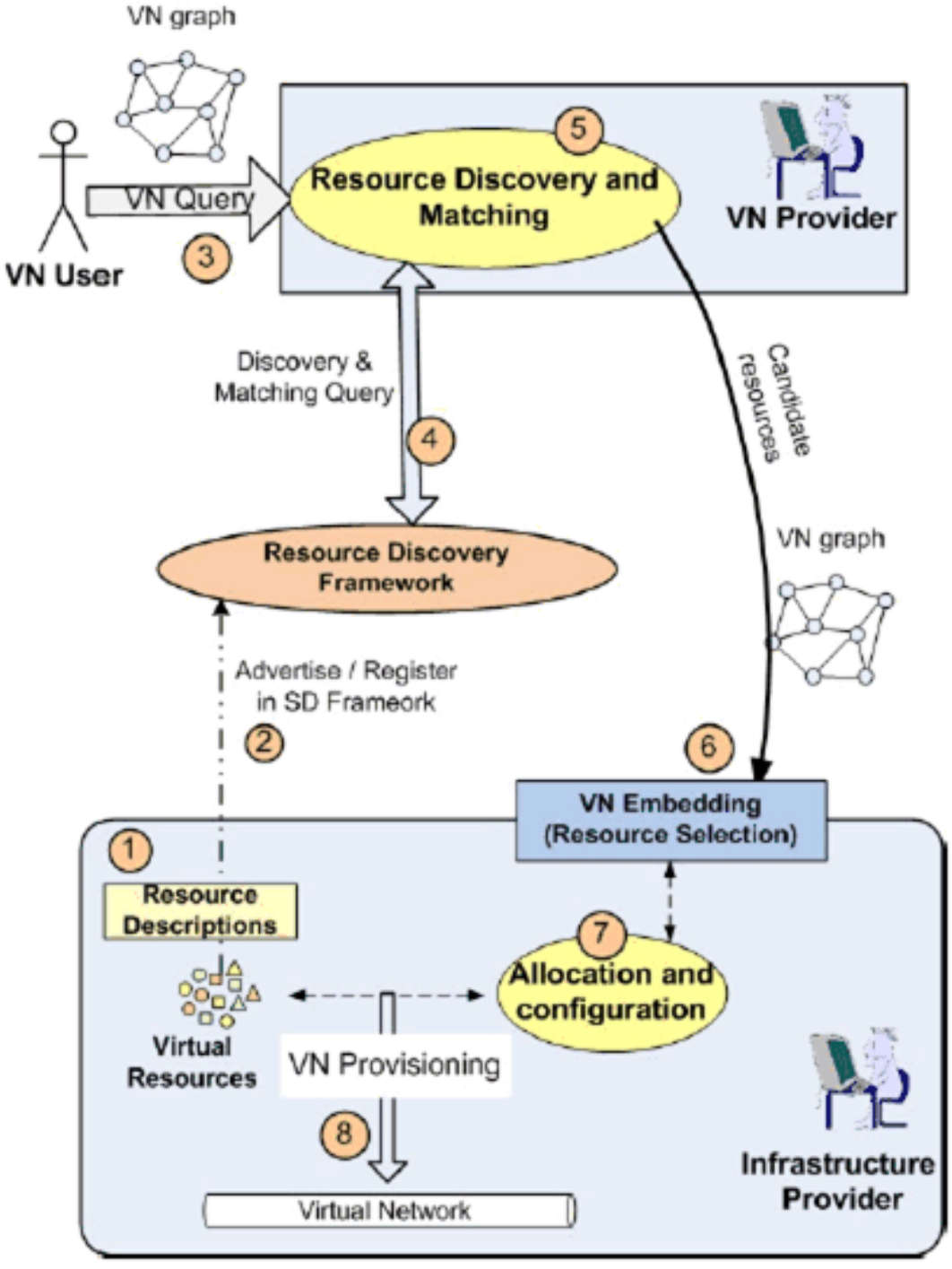

Figure 3.2: Virtual Network Embedding Framework [29].

to the online, however, it is less practical.

- **Specific VN Topologies vs. Arbitrary VN Topologies**
  Some previous work focused on a specific VN topology [38], such as the star topology, while others focused on finding the optimal VN embedding solution for any arbitrary VN topology [62].

- **Limited Resources vs. Unlimited Resources**
  The underlying substrate network resources are limited [53,62]. In order for the SP to offer a guaranteed QoS, the SP must not overbook the underlying resources (i.e. the demand should be less than the supply). The SP must perform admission control techniques to guarantee sufficient resources to satisfy the new VN request requirements. Once sufficient resources are found, the SP reserves and dedicates these resources throughout the VN lifetime. This phenomenon might contribute to some VN requests being either rejected or simply postponed. However, in some proposed work, infinite underlying resources has been assumed [38, 66], to avoid the admission control complexity. The authors assume that the SN has unlimited cpu, bandwidth or both.

- **Two-stages vs. Single-stage Mapping**
  There are two main approaches proposed in mapping VN requests: The two-stage mapping and the single-stage mapping. Most authors proposed mapping the VN request using the two-stage mapping [8, 11, 38, 53, 62, 66] while only few recommended the single-stage mapping [8, 37]. The two-stage mapping refers to first, the algorithm maps all the virtual nodes of the VN request onto the candidate substrate nodes. Next, all the corresponding virtual links are mapped onto the substrate paths. The two-stage technique simplifies the VN mapping problem. In the single-stage mapping, both the virtual nodes and the virtual links of any single VN request are mapped concurrently. Single-stage mapping technique offers better resource utilization, however, it involves huge network overhead.

- **Static Mapping vs. Dynamic Mapping**
  In static mapping, once the VN request has been mapped onto the substrate network, the allocated physical resources will be fixed irrespective of the actual usage throughout the VN lifetime. In the dynamic VN mapping, the VN requirements are dynamically adjusted to adapt to some external conditions.

- **Other Assumptions** Other assumptions were also introduced that are problem specific like allowing path splitting [62], migration [62], and re-optimization of previously

mapped VNs [20].

Work on resource allocation can be categorized as: Centralized mapping scheme and distributed mapping scheme.

3.5.3 Centralized Mapping Scheme

In the centralized mapping scheme, there is only one single entity most probably the SP that has a global view of the substrate network. The centralized entity is in charge of receiving the VN request, performing resource allocation, managing the VNs and finally tearing down the VNs once they get expired.

Multi-Commodity Flow Based Approach

Szeto et al. [53] considered the **offline** VN embedding problem with **limited** resources. In this work, the authors represented their underlying physical network as a directed graph. The virtual nodes of the VNs are assumed to be already mapped, and the objective is to minimize the cost of mapping the virtual links while ensuring the maximum resource utilization. The proposed work uses the maximum-concurrent flow problem that is a variation of the multi-commodity flow (MCF) problem.

Load Balancing

One of the earliest proposals for the VN embedding problem is that of Zhu et al. [66], which focused on achieving a low and balanced resource usage of the substrate network while assuming **unlimited bandwidth** for the SN. In their work, they introduced two new terms *node stress ratio* as the total number of virtual nodes that are mapped onto a single substrate node, and the *link stress ratio* as the total number of virtual links whose substrate path is mapped onto a single substrate link. Their objective is to keep a low and balanced stress across the whole substrate network (i.e. node and link) to avoid hot spots, reduce congestion and achieve better resource utilization. Zhu et al. argued that the combination of the previously mentioned two terms achieves a balanced load performance onto the substrate network. The authors also proposed subdividing the

VN requests into smaller graphs and map them individually to reduce the embedding problem. Selective **reconfiguration** of previously mapped VNs is also proposed to efficiently use the underlying substrate resources.

Constraint-Based Algorithm

Lu et al. [38] proposed an iterative **offline** algorithm to embed the VN requests. In this work, the authors assume that all the VNs are constrained to only one specific topology, **backbone-star topologies**. In the backbone-star topology there are two types of nodes, the backbone node that represents the center of a star, and the access nodes that represent the leaves of the backbone node. The backbone nodes are arbitrary connected to each other to form a complete graph, a ring or a star. The proposed algorithm depends on an iterative method for the VN request mapping. First, an initial mapping of the backbone nodes is performed. Next, through some iterative techniques the access nodes are connected to the backbone nodes.

The VN resource allocation theme in Lu's work,is based on **constraint-based network design**, where the resource requirements are defined in terms of a set of traffic constraints: *termination traffic constraints* that describe the total incoming and outgoing traffic from each access node, *pairwise traffic constraints* describe the traffic between two access nodes, and *distance traffic constraints* that describe the traffic flow outside the neighborhood of a node.

Path Splitting and Migration

Yu et al. [62] introduced an **online** embedding algorithm that assumes **limited resources** and avoids resource overbooking by performing admission control. The proposed VN embedding algorithm maps the VN request in **two-stages**. First, all VN requests arriving within the same time unit are collected. Second, using a greedy node mapping algorithm all the virtual nodes of the collected VN requests are mapped. Next, the k-shortest path algorithm is used to map the corresponding virtual links. The Multi-commodity Flow Problem (MFP) is used to map the corresponding virtual link onto multiple paths, using the **path splitting** ratio.

It is worth noting that, allowing multiple path have many benefits, first it allows a VN

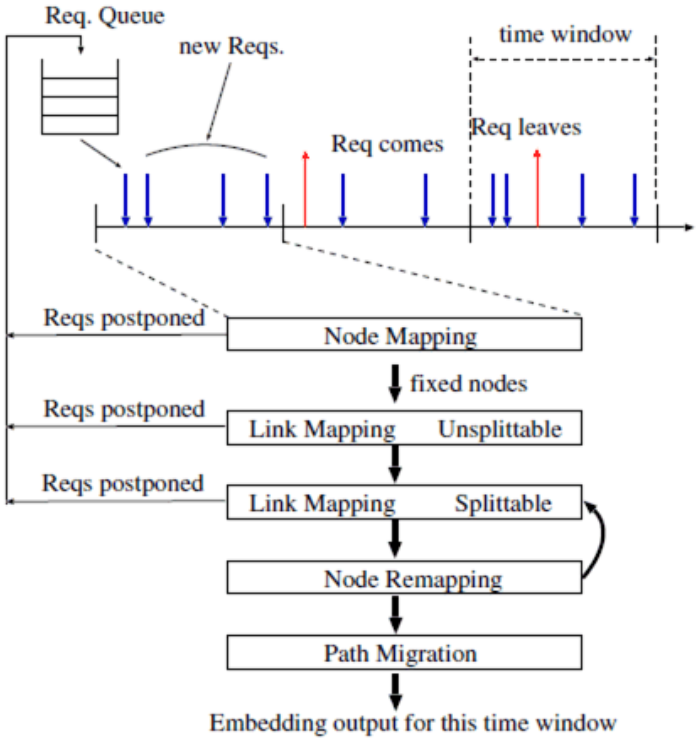

Figure 3.3: Virtual Network Provisioning [62].

request that would otherwise be rejected to be mapped, second it offers a faster recovery in case of a link failure with minimal disruption, since the traffic can be switched onto the other path(s). Fig. 3.3 is an illustration of their proposed algorithm. The authors in [59] have studied the efficiency of splitting versus single paths in routing in IP networks and showed that the performance gains of using multi-paths is bounded by the number of links in the network.

Yu et al. [62] also supported **path migration** on previously mapped VN requests to efficiently utilize the underlying resources and to allow more VN requests to be accepted. Path migration can be performed by either changing the splitting ratio or searching for new substrate paths. The authors argued that path migration should be performed on VNs with a long life span due to the encountered delay from switching the traffic onto the new path(s). Node remapping is also proposed in this work for the splittable virtual link, however, from their simulation results, its effect has shown to be very modest in the revenues compared to the computational time.

Coordinated Node and Link Mapping

Mapping both the virtual nodes and their adjacent virtual links within one **single-stage** was first attempted by Chowdhury et al. [11]. The authors' objective is to coordinate the virtual node mapping phase with its directly connected links within one stage. The substrate network was extended to an *augmented substrate graph*. A meta-node is created for each virtual node and is connected to a cluster of candidate substrate nodes through *meta-edges* with infinite bandwidth. The authors then used mixed integer programming (MIP) formulation to solve the VN embedding problem. The multi-commodity flow algorithm is then employed to map the virtual links, or the k-shortest path algorithms when path splitting is not permitted. Although the authors presented a coordination between the node and link mapping, however, creating all the meta-nodes in the first phase of the VN mapping, made the VN mapping problem fall in the **two-stage** approach.

Window Based VN embedding

Chowdhury et al. [9] proposed *ViNEYard*, a VN mapping algorithm. The algorithm defines *WiNE*, an extended online window-based VN embedding algorithm with lookahead capabilities. In their work, the algorithm queues all the arrived VN requests within a certain time window. At the end of the time window all collected VN requests are sorted based on their revenues. The VN request with the highest revenue is selected to be mapped first. If a VN request cannot be mapped within the specific time window due to insufficient resources, the request is deferred to the next time window. The authors also defined t_w, which is the maximum allowed waiting time a VN request can stay in the queue before it gets rejected. The authors use the algorithm described in [11] for the VN mapping. Although the proposed work presents an online VN embedding scheme, the selective VN mapping implicitly switch the problem to the *offline* mapping approach.

Topology-Awareness

Butt et al. [20] argue that topology-awareness and differentiating among the substrate resources can lead to higher VN acceptance ratio. The authors proposed **Re-optimization** the previously embedded virtual nodes and links that are causing a new VN request to be rejected. The depletion of the critical underlying network resources such as bridges

might have negative impact on the future VN acceptance ratio as the substrate network becomes fragmented. The authors claimed that, the optimum choice of the VN embedding will be the one that uses the least critical resources.

In order to differentiate between the underlying substrate resources, the authors introduced the *scaling factor (SF)* term. SF is defined as the likelihood of a single substrate resource to be a bottleneck. SF is a combination of another two weight factors: *Critical Index (CI)* and *Popularity Index (PI)*. CI is a fraction that measures for each substrate resource (i.e. node or link) the likelihood of the substrate network partition due to its unavailability. The CI is updated whenever the underlying network size is modified. The PI is defined as the average residual resource for a specific resource (i.e. node/link), as well as the number of VNs that are mapped on that resource.

The proposed algorithm first detects a list of all the virtual nodes and virtual links that could not be mapped. Next, a set of corresponding bottleneck substrate nodes and links are selected. The algorithm migrates these bottlenecks nodes/links to allow for the rejected VN request to be successfully mapped.

Cheng et al. [8] proposed two new VN embedding algorithms: *RW-MaxMatch* and *RW-BFS*. The proposed algorithms are based on the topology-aware resource ranking of a node. The algorithms define a value called NodeRank, where the rank of each node is affected by the node's CPU, the sum of all the bandwidth of its outgoing links and the topological characteristics of its specified neighborhood nodes.

The *RW-MaxMatch* algorithm maps the VN request in **two-stages**. First, the algorithm computes the NodeRank for the substrate network and stores it in the set m in a decreasing order based on the value of the NodeRank. The NodeRank for the VN request is also computed and sorted in the set n in a decreasing order. Next, the algorithms maps the first virtual node in n to the corresponding first substrate node in m, followed by mapping the remaining of the virtual nodes in the same manner one at a time. If the corresponding substrate node in m has insufficient cpu, then the VN request is rejected. Upon mapping all the virtual nodes, the algorithm uses the k-shortest path for unsplittable paths or the multi-commodity flow problem with splittable path to map the corresponding virtual links.

The *RW-BFS* algorithm maps the VN request in **single-stage**. In the proposed algorithm, first the NodeRank for both the substrate network and the VN request is com-

puted. Next for each virtual node a list of candidate substrate node is selected. Finally, each virtual node is embedded to one of the substrate node in its list that has sufficient cpu and satisfies the shortest path algorithm in a breadth-first search manner. In case a virtual node cannot find a suitable match from its list, the algorithm backtracks to the previously mapped virtual node and re-maps onto a different candidate, and continues mapping the remaining resources.

Subgraph Matching Algorithm

Lischka et al. [37] proposed the **online** *vnmFlib* algorithm that solves the VN mapping problem by using a backtracking algorithm based on subgraph isomorphism search method. Lischka models both the substrate network and the virtual network as a directed graph with the objective of mapping both the virtual nodes and virtual links within a **single-stage**. When a bad mapping is detected, the algorithm backtracks to the last valid mapping. During the mapping, the algorithm eliminates the substrate nodes with insufficient cpu and restricts the path length, to optimize the search space and the expensive mapping.

Network virtualization is highly dynamic (i.e. VNs are frequently added, expired, resized or moved), thus having an online single entity is very crucial. A single entity will have a global knowledge, however, it might suffer from single point of failure and scalability may be an issue. The likelihood of a centralized entity getting overwhelmed may lead to poor scalability and results in long delays before a request is accepted.

3.5.4 distributed mapping scheme

Unlike the centralized approach, the resource allocation and management responsibilities are distributed over small entities (i.e. substrate nodes). A constant communication and knowledge sharing is required between these entities to reach near optimal VN mapping results. The physical nodes need to be continuously updated to maintain up to date information about the underlying network resources.

Multi-Agent Based Approach

Houidi et al. [28] proposed a dynamic and distributed VN mapping algorithm that relies on the *Multi-Agent* based approach [54]. The scheme assumes **unlimited** underlying resources and addresses the **offline** version of the problem. The proposed algorithm decomposes the VN request into smaller interconnected hub-and-spoke clusters to reduce the mapping complexity. Each substrate node of the substrate network runs three distributed algorithm in parallel in order to map the VN request:The *Capacity-Node-Sorting* algorithm, *Shortest Path (SPT)* Algorithm, and the *main VN mapping* algorithm.

In the *Capacity-Node-Sorting* algorithm, the substrate nodes continuously exchange the substrate network available capacity. Each substrate node constructs and maintains a capacity vector that represents the available capacity of all the substrate nodes. The vector is sorted based on the nodes' capacity in a decreasing order. The collected data is communicated to the main VN mapping algorithm at runtime of a VN request getting mapped.

The *SPT* algorithm each node computes and stores a set of the shortest paths from itself to all other substrate nodes and its associated weights by using the distributed Bellman-Ford algorithm.

In the *main VN mapping* algorithm, the algorithm executes the *Hub-Spokes-Mapping* procedure for the mapping the VN request. The procedure searches for the cluster whose hub (i.e. center node) has the highest cpu, and select it to be mapped first, the cluster is then removed from the VN request. To map the cluster, the substrate node with the highest cpu is selected to map the hub node, and based on the SPT algorithm, its corresponding spoke nodes are then mapped. Once the cluster is mapped, the process of mapping the next cluster is repeated again until the whole VN topology is mapped.

Ant Colony

Fajjari et al. [17] proposed the *VNE-AC* VN embedding strategy that is based on the Max-Min Ant System [52] meta-heuristics. The proposed algorithm is divided into the following stages: First, the virtual access nodes of the VN request with specific geographic constraints are mapped. The corresponding virtual links connecting the virtual access nodes are then mapped. Next, using an iterative approach, the authors split

the remaining unmapped virtual nodes and links into small components known as the *solution components* and sort them in order based on the component with the highest number virtual links with their outermost virtual nodes are already mapped.

The proposed algorithm uses parallel artificial ants iteratively to explore the search space for available candidates within a certain allowed number of hops from the previously mapped virtual nodes. The algorithm maps the small components one at a time onto the potential candidates. Based on some desired objectives in selecting the appropriate candidates, such as substrate nodes with highest available resources or substrate links with highest available bandwidth, and the associated pheromone trail, the potential candidates are selected. Finally, the algorithm selects the VN mapping with the minimum mapping cost, and the pheromone trail gets updated.

Adaptive VN Provisioning

Houidi et al. [29] extended the VN embedding problem [28] to include the Adaptive VN provisioning. The adaptive VN provisioning is mainly concerned with the dynamic changes that might occur in the VN or the substrate network such as changes in service demand, resource failure or severe performance degradation, and how to preserve the mapped VNs' service level agreement. In this work, Houidi et al. proposed a distributed fault-tolerant VN embedding algorithm. The proposed algorithm relies on the multi-agent based framework for the distributed coordination (e.g. negotiation and synchronization). The autonomous agents are integrated in the substrate nodes that are continuously communicating and exchanging information. In case of any failure, the agents decide locally the reselection process.

Fig. 3.4 illustrates the proposed adaptive VN embedding scheme. The right side of the figure represents the initial VN mapping [28], and the left side represents the proposed adaptive VN provisioning. If a change in the VN resources is required (i.e. *Case 1*), such as adding or removing some resources, the adaptive algorithm search for the new candidates and updates the VN. If a resource failure occurs, such as in *Case 2*, the distributed adaptive fault-tolerant embedding algorithm detects the failure and allocates new resources to replace the failed ones. The algorithm also detects the congested or overloaded links on the substrate network and reassigns the affected virtual links onto different paths.

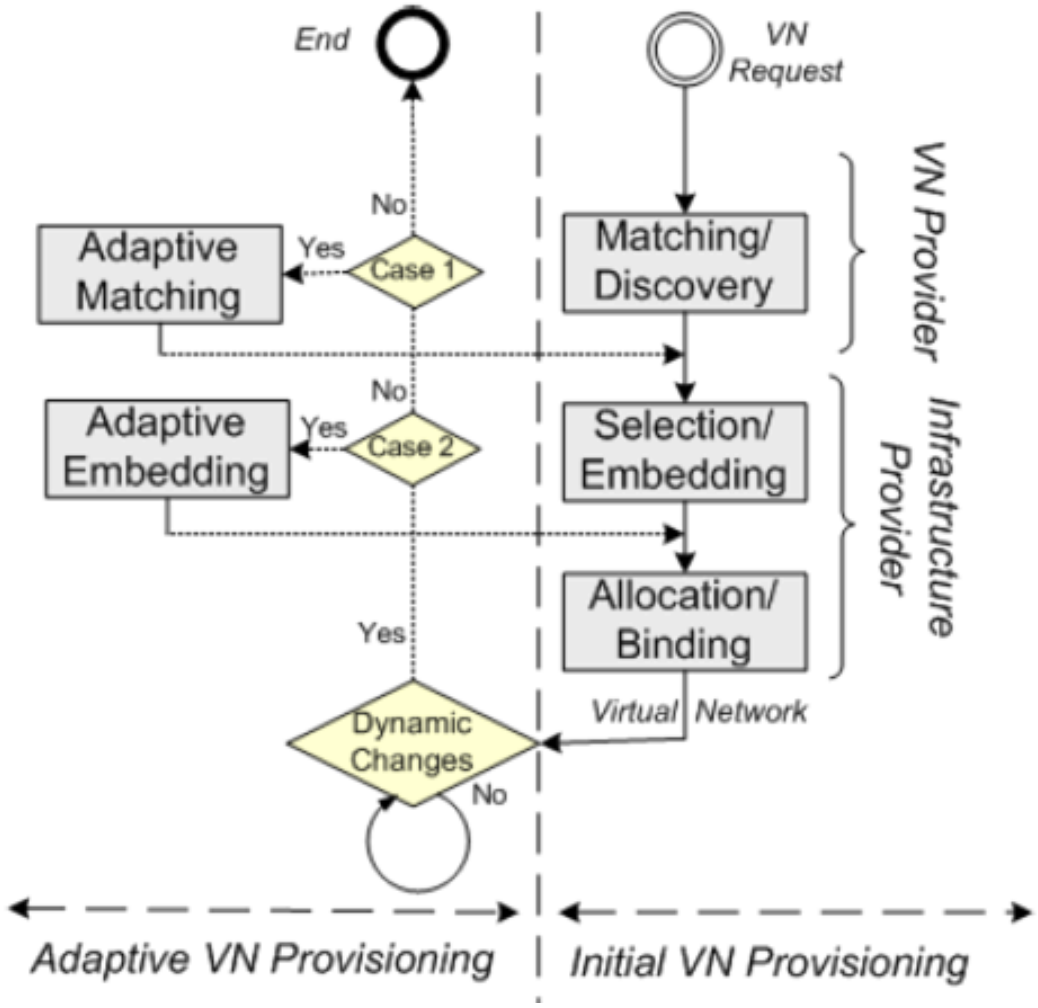

Figure 3.4: Adaptive VN Embedding [29].

Policy-Based End-to-End Embedding

Chowdhury et al. [10] argued that in practical, the VNs are mapped across multiple InPs. InPs have no accessible information on how each InP is mapped over a particular segment of the VN request. However, all the InPs' objective is to get as much as they can from the VN embedding that will lead to higher revenues while minimizing their expenditure. The authors introduced *PolyViNE*, a policy-based end-to-end inter-domain VN embedding protocol that is based on the geographic constraints. PolyViNE allows each InP to impose its policy and embedding algorithm within its network and have a mutual agreement between other InPs.

The proposed algorithm creates a highly competitive market within InPs by allowing bidding at every level of VN embedding. PolyViNE assumes that all the virtual nodes of the VN request have some location constraints that need to be satisfied, and thus the virtual nodes are sent to the appropriate InPs. The SP sends a message with the VN request or the appropriate partition to multiple InPs and waits for their reply.

If an InP is not capable of mapping part or all of the VN request due to admission control or lack of profitability, it may outsource the request to other InPs to complete the embedding of the VN request or replies back to the SP with a rejection message. Whether the VN request is mapped or rejected, the InP will reply back to the SP with either its rejection reasons, or its successful embedding cost and a list of other InPs involved in the embedding. Finally, the SP selects the VN embedding with the lowest price.

Game Theory

Zhou et al. [65] introduced the Game Theory as a mean for economic incentive efficient bandwidth allocation theme. Their adopted model consists of one InP and multiple VNs, and their focus is on how InP can allocate its limited resources to multiple VNs in a distributed non-cooperative game model. The scheme assumes that the total amount of bandwidth of the participating VNs exceeds the SN capacity.

The authors proposed a pricing scheme that is only enforced by the InP. The payoff function of each VN request involves: The *Utility Function*, the *Pricing Function* and the *Congestion Function*. The utility function depends on both the bandwidth allocated to the VN request and the path rate. The pricing function is the assigned InP price for the specific VN request. The congestion function is based on the assigned bandwidth and the path rate. The VN objective is to increase its revenues by increasing its path rate, however since congestion on certain links may decrease its revenues, VN may choose to decrease its path rate at the congested links.

VN provisioning

Several approaches have been proposed to deal with VN changes. The authors in [43] showed how self-organization can be employed to migrate the virtual nodes from one

substrate node to the other. Another example of VN provisioning is that of the DaVinci architecture [27] where authors focus on efficient scheduling of the substrate link bandwidth among its already hosted virtual links according to a given VN utility. Although dynamic scheduling provides an efficient means for utilizing the substrate resources, it does not provide hard resource guarantees for individual VNs.

3.6 Discussion

Employing a distributed algorithm to embed the VNs overcomes the bottleneck problem of the centralized schemes but lacks a global view of the underlying SN resources leading to less efficient utilization of the resources and incurring a much higher message exchange cost. Another interesting observation is that the aforementioned approaches mostly rely on performing a network-wide search to find one mapping solution for VNs that are much smaller in size than the SN regardless of the current load of the network. While the majority of the VN requests can be satisfied by searching for a good solution in a smaller sub-domain within the network, as will be illustrated by our experimental results. The proposed embedding scheme focuses on addressing the aforementioned two limitations.

In Chapter 4, we propose a novel hierarchical substrate management framework that performs concurrent searches for more than one efficient VN mapping solution from which the best solution is selected. The proposed architecture achieves load-balancing through partitioning the underlying substrate into a number of management domains; these domains vary in size at each level of the hierarchy allowing for efficient mapping of VN requests of various sizes. The architecture also achieves a balance between the single manager bottleneck in centralized schemes and the message overhead incurred in fully distributed ones. Furthermore, we introduce a novel VN mapping algorithm based on an exact subgraph matching [5]. Since, this technique can only perform exact link-to-link mapping, rather than a link-to-path mapping, the proposed algorithm circumvents this problem by dynamically reshaping specific parts of the network.

In Chapter 5, we propose a novel time-varying VN model as well as a new pricing mechanism for the VNs that together efficiently maximize the utilization of the SN resources. We then develop a novel distributed embedding scheme that is tailored to

the characteristics of the new time-varying VN requests. To the best of the authors' knowledge, the presented work is the first to incorporate a pricing mechanism and a time-varying demand model as an incentive for the efficient management of the SN resources. A summary of some of the approaches for VN embedding problem is presented in Table 3.1.

Work	**On/ Off**	**Approach**	**Resources**	**Stage**
Lu et al. [38]	offline	- Heuristics - Specific VN topology - Traffic constraints	-Unlimited bandwidth	Two-stages
Zhu et al. [66]	offline	- Heuristics - Node/link stress	Unlimited	Two-stage
Yu et al. [62]	online	- Heuristics - Path splitting - Migration	Limited	Two-stage
Chowdhury et al. [11]	online	- Mixed integer programming - Coordinated node & link mapping	Limited	Two-stage
Lischka et al. [37]	online	- Subgraph isomorphism detection	Limited	Single-stage
Houidi et al. [28]	offline	- VN decomposition - Shortest path algorithm	Unlimited	Two-stage
Szeto et al. [53]	offline	- Only finding path - MCF problem	Limited	Two-stage
Butt et al. [20]	online	- Re-optimization - Resource differentiation	Limited	Two-stages
Cheng et al. [8]RW-BFS	online online	- RW-BFS algorithm - RW-MaxMatch algorithm - Node ranking - Shortest path for unsplittable path -MCF for splittable path	Limited Limited	single-stages Two-stages
Chowdhury et al. [9]	offline	- Window-based VN mapping - Coordinated node & link mapping	Limited	Two-stage
Fajjari et al. [17]	online	- Max-Min Ant System metaheuristics - VN decomposition - Parallel search	Limited	Two-stage
Chowdhury et al. [10]	online	- Policy-based VN embedding across multiple InPs - Location constraints	Limited	—

Chapter 4

VN Embedding in a Hierarchical Management Framework

4.1 Introduction

In this chapter, a novel hierarchical substrate management framework is presented that selects the optimum VN mapping solution from a variety of simultaneously searched possible efficient VN mapping candidates. Based on the size of the SN and the size of the VN the number of available VN mapping candidates are evaluated as discussed later in this chapter. The presented hierarchical management framework achieves load balancing and has higher VN mapping acceptance ratio compared to the flat SN, as will be later seen in Section 4.6, by selectively choosing the optimum VN mapping candidate.

To ensure efficient resource allocation to the VNs, the SP needs to guarantee the following:

- Efficient management and maintenance of the underlying infrastructure.
- The SP hosts and accommodates a variety of VNs that offer a variety of different network services and applications. Each VN imposes different resource constraints that need to be satisfied. These network applications might be as simple as file transmitting or as complex as voice and video interaction transmission.

- Efficient resource utilization while not sacrificing the guaranteed Quality of Service (QoS) to the end-user. The expected QoS differ from one VN to the other based on the service it offers.

4.2 Chapter Organization

The remainder of this chapter is organized as follows:

- The network model and the VN mapping problem are discussed in Section 4.3.
- An example of a simple VN embedding problem is discussed in Section 4.4.
- Section 4.5 discusses the proposed VN mapping scheme and provides details of the created hierarchical substrate management framework.
- Section 4.6 is dedicated for the experimental results of the proposed scheme and the performance evaluation.
- Finally Section 4.7 concludes the chapter.

4.3 Network Model and Problem Formulation

4.3.1 Network Model Definition

The SN is represented by an undirected graph $G(N, L, \boldsymbol{c}, \boldsymbol{b})$ where N and L represent the sets of substrate nodes and substrate links, respectively. Associated with each node $n_s \in N$ is a value c_s to represent its available processing capacity. Similarly, associated with each link $l_{sd} \in L$, that is between n_s, $n_d \in N$, a value b_{sd} to reflect its available bandwidth capacity. The vectors $\boldsymbol{c}$ and $\boldsymbol{b}$ are used to maintain the capacities of the SN nodes and links, respectively. A simple routing path k between any two nodes n_s, n_d $\in N$, denoted as $p_k(n_s, n_d) \subset L$, defines a unique sequence of links where no node is traversed more than once in p_k. The set of all simple routing paths within a specified number of hops is denoted by P.

A VN request represents a description for a specific service or application to be performed using the resources of the SN. It specifies a number of tasks to be executed with CPU processing requirements for their execution and the bandwidth needed for their communication along with the request lifetime [60]. More precisely, a VN request from user $x \in \{1, \cdots, |\mathcal{X}|\}$, where $\mathcal{X}$ is the set of subscribers to the VN services, can also be modeled by $G^x(N^x, L^x, \boldsymbol{c}^x, \boldsymbol{b}^x)$ where N^x and L^x are the sets of virtual nodes and links in G^x, respectively. The vectors $\boldsymbol{c}^x$ and $\boldsymbol{b}^x$ maintain the requested VN processing cycles and bandwidth, respectively. Each node $n_i^x \in N^x$ is associated with the requested processing resources c_i^x and each link $l_{ij}^x \in L^x$, between nodes $n_i^x, n_j^x \in N^x$ is associated with a requested bandwidth b_{ij}^x. Each VN request, G^x, has a desired duration τ_x which represents the expected VN lifetime after embedding.

The VN embedding problem involves finding two *one-to-one* mappings: $\mathbf{M}^{\text{nodes}} : N^x \rightarrow N$, and $\mathbf{M}^{\text{links}} : L^x \rightarrow P$ defined as follows.

$$\begin{aligned} \mathbf{M}^{\text{nodes}}(n_i^x) &= n_s, \text{ s.t. } R(n_s) \geq c_i^x, \\ \mathbf{M}^{\text{nodes}}(n_i^x) &\neq \mathbf{M}^{\text{nodes}}(n_j^x), \forall n_i^x, n_j^x \in N^x, i \neq j, \end{aligned} \tag{4.1}$$

where $R(n_s)$ is the residual capacity of SN node n_s after subtracting from c_s the CPU cycles already allocated to other active VNs. In a similar manner, $\mathbf{M}^{\text{links}}$ is defined as follows.

$$\mathbf{M}^{\text{links}}\left(l_{ij}^x\right) = p_k\left(\mathbf{M}^{\text{nodes}}(n_i^x), \mathbf{M}^{\text{nodes}}(n_j^x)\right) \tag{4.2}$$

where $p_k\left(\mathbf{M}^{\text{nodes}}(n_i^x), \mathbf{M}^{\text{nodes}}(n_j^x)\right) \in P$ is a unique path such that $b_{ij}^x \leq R(l_{sd})$, where $R(l_{sd})$ is the residual bandwidth of each SN link $l_{sd} \in p_k\left(\mathbf{M}^{\text{nodes}}(n_i^x), \mathbf{M}^{\text{nodes}}(n_j^x)\right)$ after subtracting from it the amount of bandwidth already allocated to other links of active VNs in addition to bandwidth already allocated other links in the VN request of user x. Herein, $\mathbf{M}^{\text{nodes}}(N^x)$ and $\mathbf{M}^{\text{links}}(L^x)$ will be used to refer to the sets of substrate nodes and paths resulting from mapping VN nodes in N^x and links in L^x, respectively.

4.3.2 Problem Formulation

The objective of the SP is to maximize its profit $\mathbf{P}(G^x)$, through finding the optimum mapping of a VN request onto the SN. The SP profit is basically the difference between what the VN subscriber is willing to pay for the requested resources i.e. $R(G^x)$, and the

SP cost, i.e. $\mathbf{C}(G^x)$, encountered in dedicating the requested resources during the VN lifetime. However, it should be emphasized that the VN subscriber is only willing to pay for the required virtual resources and will not pay for any additional resources reserved by the SP. Hence, the objective of an efficient VN mapping is to minimize the incurred $C(G^x)$.

More precisely,

$$P(G^x) = R(G^x) - C(G^x) \tag{4.3}$$

such that,

$$R(G^x) \;=\; \rho^{\text{cpu}} \sum_{n_i^x \in N^x} c_i^x + \rho^{\text{bw}} \sum_{l_{ij}^x \in L^x} b_{ij}^x \tag{4.4}$$

and

$$C(G^x) \;=\; C(\mathbf{M}^{\text{nodes}}(N^x), \mathbf{M}^{\text{links}}(L^x)) \tag{4.5}$$

$$\begin{aligned} C(G^x) = \tau_x \Big(& \omega_{\text{cpu}} \sum_{n_i^x \in N^x} c_i^x \\ & + \omega_{\text{bw}} \sum_{l_{ij}^x \in L^x} |p_k(\mathbf{M}^{\text{nodes}}(n_i^x), \mathbf{M}^{\text{nodes}}(n_j^x))| b_{ij}^x \Big) \end{aligned} \tag{4.6}$$

where ρ^{cpu} and ρ^{bw} are the prices of a unit of substrate node processing cycles and link bandwidth, respectively. The operation $|.|$ in (4.6) (also referred to as the objective function) returns the length (in hops) of the path $p_k \in \mathbf{M}^{\text{links}}(\mathbf{L}^x)$ and ω_{cpu} and ω_{bw} are constant factors that depend on the SP's operational costs of managing the SN nodes and links, respectively.

The first part in (4.6) is fixed, since there is only one-to-one node mapping allowed, however, the second part of the equation depends on the path length. Thus the objective of the VN embedding is to find the mapping with the least path length.

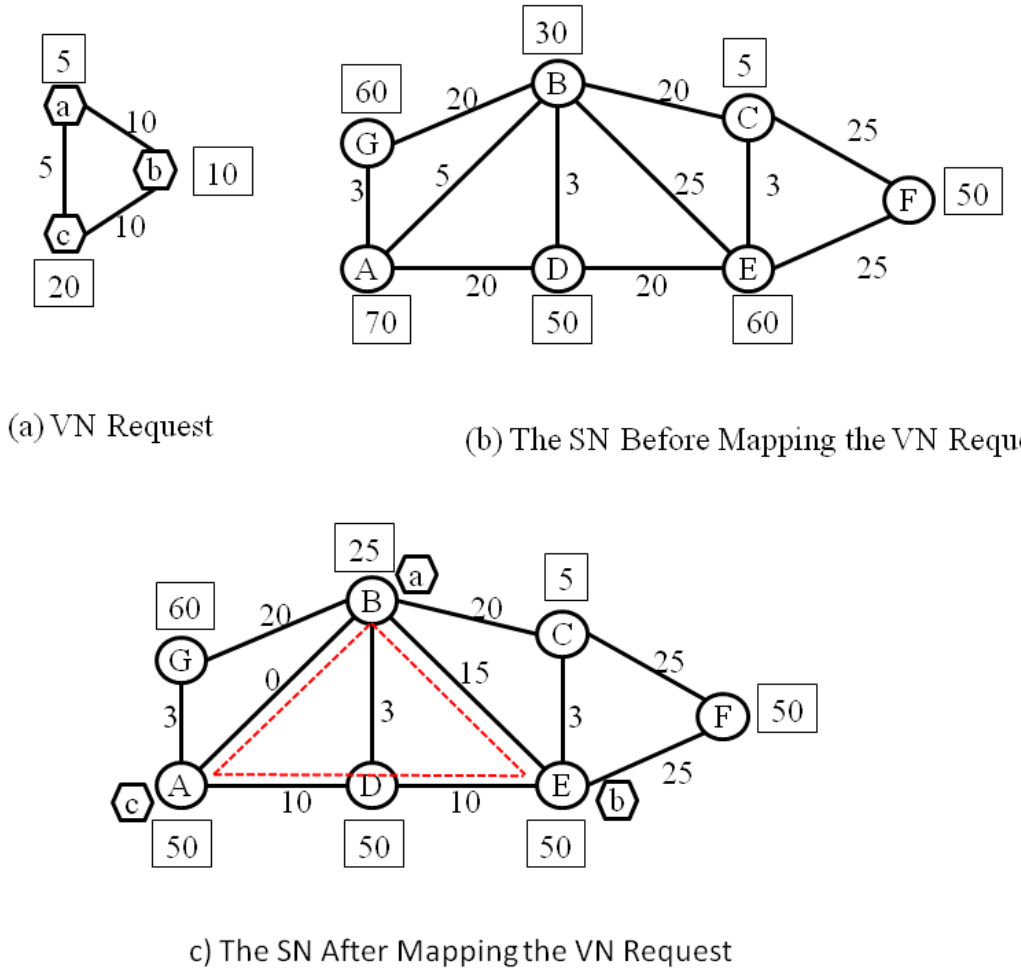

Figure 4.1: Example of a 3-level hierarchy of VMs.

4.4 Example of VN Embedding

Fig. 4.1 illustrates a simple example of a VN embedding. The VN request is represented by the undirected graph G^x as shown in Fig. 4.1(a). The VN request consists of 3-nodes (i.e. $N^x = (a, b, c)$), and 3-links (i.e. $L^x = ((a, b), (b, c), (a, c))$). The value in each square i.e. node, and on the top of each line, i.e. link, represents the cpu and the bandwidth constraints, respectively.

Similarly, Fig. 4.1(b), represents the undirected graph G that consists of 7-nodes $N = (A, B, C, D, E, F, G)$, and 11-links
$L = ((A, B), (A, D), (A, G), (B, C), (B, D), (B, E), (B, G), (C, E), (C, F), (D, E), (E, F))$.

The value in each square represents the available cpu for each specific node n_s, i.e. $R(n_s)$, and the value on each link l_{sd} represents the current available bandwidth, i.e. $R(l_{sd})$.

Fig. 4.1(c), shows the embedding of the VN request on the SN, such that $\mathbf{M}^{nodes}(N^x)=(B, E, A)$ and $\mathbf{M}^{links}(L^x)=((B, E), (E, D, A), (B, A))$. The values of the cpu and bandwidth of the SN reflect the current residual resources, $R(n_s)$ and $R(l_{sd})$, respectively. Both (a, c) and (a, b) virtual links are each mapped onto one single substrate link, while (b, c) is mapped onto the $p_k = (E, D, A)$ due to insufficient resources. The reserved physical resources are dedicated to the VN throughout its lifetime. Once the VN expires or the resources is not needed, the SP restores its resources.

4.4.1 Subgraph Matching Problem

The subgraph matching problem, often referred to as *the subgraph isomorphism problem*, is a form of exact pattern matching that have been intensively studied due to its various applications in the analysis of chemical structures, pattern recognition, representing complex structures like the Chinese characters [44], and others.

The subgraph matching problem can be simply represented by two undirected graphs, $G^1(N^1, E^1)$ and $G^2(N^2, E^2)$. The exact subgraph matching problem is concerned with finding a one-to-one mapping $f : N^1 \rightarrow N^2$ such that: $(u, v) \in E^1 \iff (f(u), f(v)) \in E^2$ [5]. In this chapter, an adequate iterative algorithm is adopted that relies on a depth-first search with backtracking and pruning unfruitful branches (paths) [5,12], that should result in a significant reduction in the search time. In this technique, a partial match is first found between two graphs (i.e., G^1 and G^2) and added to an empty set. The algorithm then proceeds with other partial mappings until the whole graph is completely matched. In case of any mapping failure, the algorithm backtracks to the previously added partial mappings and searches for another alternative route.

4.5 The Proposed VNMA Embedding Scheme

This section is devoted to my contribution and is divided into two parts: First, a detailed description of the proposed SN hierarchical framework that reduces the VN request mapping search space is discussed in Section 4.5.1. Next, the proposed the VN Mapping

Algorithm (VNMA) is introduced and discussed in Section 4.5.2.

4.5.1 The Hierarchical SN Management Framework

As mentioned earlier, the objective of an efficient VN embedding is to maximize the SP profit (4.3) by minimizing the mapped VN path length. The proposed substrate management framework partitions the SN into a hierarchy of management sub-domains, and each sub-domain manages its subnetworks as shown in Fig. 4.2. Based on the size of the VN request and the SN, the VN request is sent to the appropriate sub-domain to be mapped. The VN is only allowed to be mapped within one single subnetwork to guarantee the locality of the mapped VN request.

The number of the virtually created hierarchical level k merely depends on the SP and the size of the SN. There are many possible ways to partition the SN. The SN partition can be based on the nature of the expected VN requests. For example, it can be performed based on the geographic location of the substrate nodes, if some of the VNs are expected to impose some geographical constraints. Another possible objective that is used in this simulation is based on the multi-constraint graph partitioning problem [32], where the objective is to minimize the edge-cut (i.e. inter-links between the sub-domains) while partitioning the sub-domains into equal size (i.e. same number of nodes in each subnetwork).

The proposed SN management framework offers a balance between the advantages of the distributed management, that alleviates the pressure from one single entity, and the advantages of the centralized management, that offers a global view of the underlying SN.

The SN management framework partitions the G into a hierarchy of k-levels virtualization managers, i.e. (VM_k)s, sub-domains. The top level of the hierarchy $h = 1$ represents the entire SN and is managed by the SP. At each level $1 < h \leq k$ down in the hierarchy, the SN is partitioned into h management sub-domains.

Fig. 4.2 depicts an example of a three-layer SN management architecture (i.e. $k = 3$). SP manages the top level $G_1 = G$, the following $h \leq k$ hierarchical levels are individually managed by their own centralized coordinator VM_h as shown in Fig. 4.2. At each level h in the hierarchy, there is one centralized virtualization manager VM_h that manages

the G_h, where G_h is defined as the set of h disjoint subnetworks, $\left(g^h_{sub}\right)$s, at the level h in the hierarchy, more formally:

$$G_h = \{g^h_1, g^h_2, ..., g^h_h\}. \tag{4.7}$$

The assigned coordinator VM_h receives the VN request from SP, next it assigns the VN request to its managed h-subnetworks to be mapped. Each VM_h monitors its available resources and performs periodic repartition of its h-subnetworks following the partitioning techniques described in [33] that is based on fill reducing orderings for sparse matrices. This operation guarantees efficient utilization of the underlying network resources as well as it avoids congestion on specific hot substrate links. The details of the SN pertaining process can be found in details in [32, 33]. Periodic repartition offers load balancing by allowing all the substrate links (i.e. inter-links and intra-links) in the sub-domain to be efficiently utilized.

The VM_h reports to the SP any physical changes occurred in its resources, and informs it of any VN request that got rejected in its sub-domain.

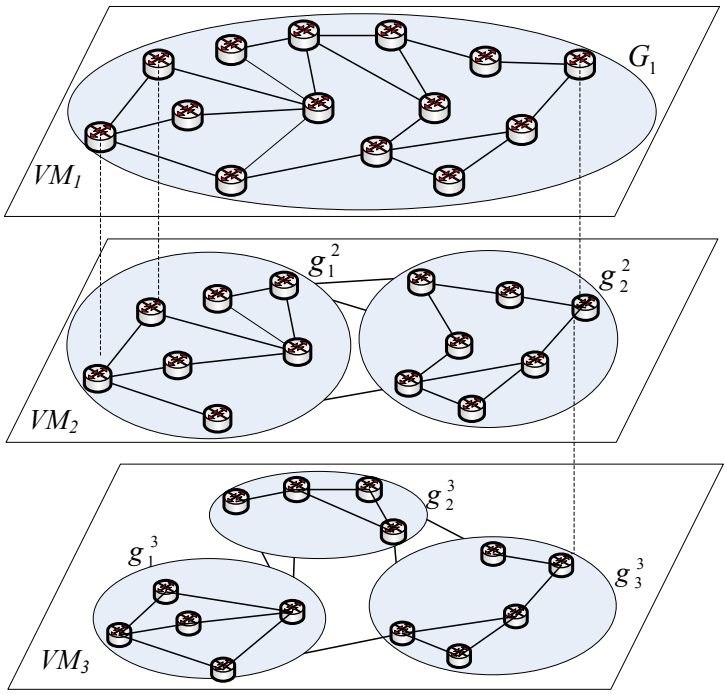

Figure 4.2: Example of a 3-level hierarchy of VMs.

As the VN requests arrive at the SP, the SP stores the VNs in a service queue and regularly dequeues one request at a time based on FIFO technique. The SP acts as the top coordinator in the hierarchy and forwards the request to the appropriate management level h. Once the SP forwards the VN request, the queue is halted until the appropriate VM_h sends an update of whether the received VN request has been mapped or rejected.

The SP selects the appropriate management level h according to the following rule:

$$h = \begin{cases} \gamma & \gamma \leq k \\ k & \gamma > k \end{cases} \tag{4.8}$$

where k is the number of levels in the hierarchy, and γ is a variable that is based on:

$$\gamma = \left\lfloor \delta \frac{|N|}{|N^x|} \right\rfloor \tag{4.9}$$

Here, $\delta \in [0.5, 0.75)$ is a constant specified by the SP. The choice of δ is very crucial as it substantially affects the search time required for the VN mapping process as well as it affects the probability of successfully mapping a VN request. The effects of different δ is discussed in more details in Section 4.6. When the SP receives the VN request from the queue, it sends it to the appropriate h^{th}-level coordinator, VM_h, as shown in Equation (4.8). VM_h decides how many and which subnetworks at its sub-domain can be used to search for an optimal VN mapping solution. The VM_h decision can be based on the strict VN request's geographic constraints or scarce resources at certain subnetworks. If there are none specified constraints, the VM_h will send the VN request to all its h-subnetworks. The search is performed in parallel to find the optimum VN mapping solution. In this work, there is not imposed any geographic location constraints for the received VN requests and all the h-subnetworks within the same hierarchical level are treated equally.

Each h-subnetwork within the same hierarchical level simultaneously searches for an efficient VN embedding that will both minimize the mapping cost, as shown in Equation (4.6), as well as achieves load balance within its subnetwork, as will be discussed in Section 4.5.2. Next, all the h-subnetworks report back their mapping result to their VM_h coordinator. The coordinator, out of all the reported mapping results, selects the subnetwork with the minimum cost to map the VN request. Finally, the VM_h notifies the SP the success or the failure of embedding the VN request.

If the VN request cannot be mapped by the selected VM_h coordinator due to insufficient resource, then the SP may decide to forward the request up in the hierarchy for the higher level $h-1$ coordinator to repeat the process. The maximum number of allowed forwarding levels have been experimentally set to 2, such that the probability of successfully mapping the VN request increases while controlling the search time. Hence, the algorithm allows the unmapped VN request to be forwarded up in the hierarchy twice before the request gets rejected and sent back to be re-queued.

4.5.2 VNMA Overview

In this section, a detailed pertinent is provided to the proposed VN mapping scheme, VNMA. The following are the main steps in the VNMA:

- The SP partitions the VN request into smaller star-shaped subgraphs, (vn^x)s, as will be discussed in details in Section 4.5.2. Next from Equation 4.8, SP assigns the appropriate VM_h-coordinator to map the request. The VM_h forwards the vn^xs to its h-subnetworks to map the request.
- Each of the star-shaped subgraph is individually and gradually mapped on the h-subnetworks in parallel, as shown in Section 4.5.2, using the proposed VNMA mapping algorithm. Finally, the h-subnetworks report their mapping cost function in Equation (4.6) to their VM_h.
- Based on the embedding results, the VM_h coordinator selects the appropriate subnetwork to map the VN request, such that, it will offer the maximum embedding profit, as shown in Equation (4.3).
- The VM_h sends an update to the SP.

VN Partitioning

SP partitions G^x into smaller star-shaped subgraphs i.e. vn^xs to accommodate different types of VN topologies and to increase the probability of a VN request being successfully mapped, by simply mapping one partition at a time. In order to partition the G^x, first the proposed algorithm, VNMA, finds the appropriate minimum vertex cover set

$\Theta^x = \{n_1^x, n_2^x, ..., n_\rho^x\} \subseteq N^x$ [4]. Θ^x is defined as the minimum vertex dominating set such that every link in L^x is incident on a vertex in Θ^x. Each virtual node in Θ^x represents the center node of the star-shaped subgraph, vn_i^x ,$i \leq \rho$,and the ρ represents the number of created subgraphs. The virtual node in Θ^x are sorted in a descending order based on their degrees, i.e., number of adjacent links.

Θ^x can be obtained via a simple heuristic as follows;

1. First, Θ^x is an empty set, and all the virtual links are untouched.
2. The virtual node with the highest degree is selected from N^x to be the first node in Θ^x.
3. All the virtual links that are adjacent to the selected virtual node are then said to be *touched*.
4. At each step, the virtual node with the next highest node degree, and has at least one adjacent link $l_{ij}^x \in L^x$ that it touched, as well as at least one untouched virtual link is selected to be the next node in Θ^x.
5. The steps of adding virtual nodes to Θ^x is repeated until there is no more untouched virtual links.

Associated with every node $n_i^x \in \Theta^x$ the corresponding star-subgraph $vn_i^x \subseteq G^x$, with n_i^x as its root and all its adjacent neighbors as its leaves, to construct the stars set $\boldsymbol{vn}^x = \{vn_1^x, vn_2^x, \cdots vn_{\rho\}}^x$.

Example of VN Partitioning

An example of the vertex dominating set is shown in Fig: 4.3. The VN request, G^x, is represented with 10-nodes and 11-links, as shown in Fig: 4.3a. The virtual node $\boldsymbol{a}$ has the highest degree in the graph, and it is selected to be the first virtual node in Θ^x. The virtual node $\boldsymbol{a}$ and its adjacent virtual nodes are the first subgraph in $\boldsymbol{vn}^x$. Next $\boldsymbol{b}$ is added to Θ^x as it has the next highest node degree, and has at least one touched link and at least one untouched link. Finally, $\boldsymbol{c}$ is also added to Θ^x and there are no more virtual nodes to be added to Θ^x, all the virtual links have been touched. The process is

then stopped. Fig:4.3b is the corresponding the VN request after it has been partitioned into 3-subgraphs, thus $\Theta^x = \{a, b, c\}$, and $\rho = 3$.

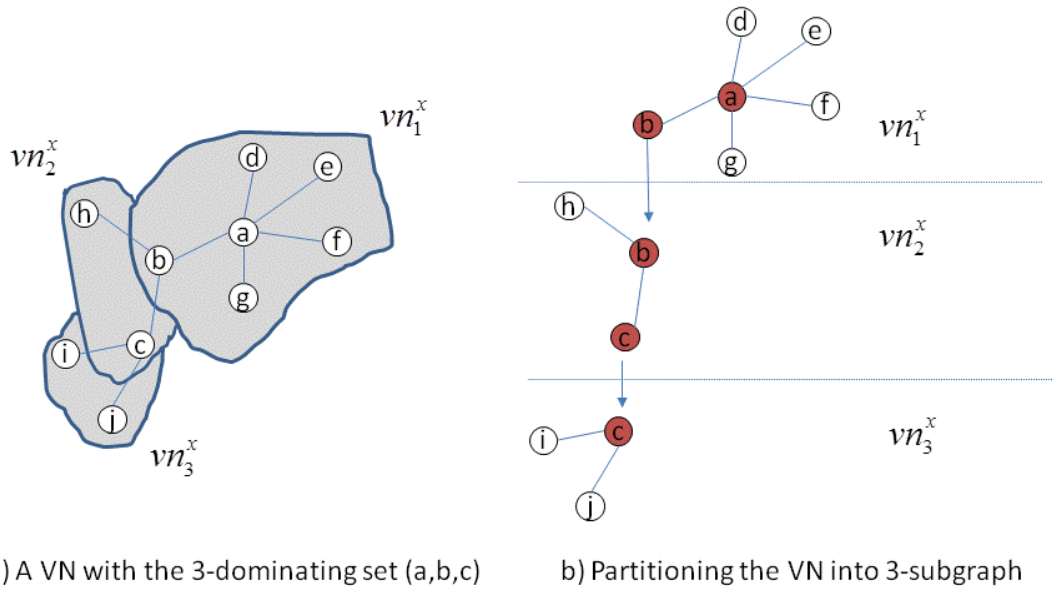

Figure 4.3: An example of VN partitioning into 3 subgraphs.

Mapping the VN Stars

All the received VN requests are stored in the SP service queue, and are individually dequeued by the SP one at a time, and sent to the selected VM_h, as discussed in Section 4.5.1, to be embedded. VM_h sends the vn_x set to all its h-subnetworks. The h-subnetworks attempt mapping the VN request using the proposed VNMA, as will be discussed later, and forward their results back to their VM_h. The proposed mapping scheme is initiated once the SP dequeues the first VN request from its queue as shown in (Algorithm 1 Line: 1).

Each subnetwork in the selected h-domain try to map the same VN request in simultaneously as shown in (Algorithm 1 Line: 8 - 10). The proposed algorithm gradually maps one single star vn_i^x at a time (Algorithm 2). The mapped vn_i^x is then removed from $\boldsymbol{vn}^x$, and the next selected vn_i^x is mapped. The selection of the next vn_i^x depends on the following:

- The $n_i^x \in \Theta^x$ with the highest node degree.
- At least one of its adjacent neighboring virtual nodes is already mapped.

Based on these criteria, in the VN example of Fig: 4.3b, vn_1^x will be mapped first as it has the highest node degree in the G^x request. Next, although vn_2^x and vn_3^x have the same node degree, but vn_2^x will be selected to be mapped prior to vn_3^x as it has at least one virtual node that is already mapped, i.e node b.

The first criteria for the next vn_i^x selection allows the VNMA algorithm to first allocate and reserve resources to the star-subgraphs that contains higher resource constraints and thus increasing the probability of the VN request to be successfully embedded. The second condition of selecting the next star subgraph guarantees that the mapping of the new star-subgraph is within the same neighborhood (i.e. maximum allowable number of hops) as the previous stars, thus achieving minimum mapping cost as discussed in Section 4.5.4.

The proposed VNMA algorithm maps vn_i^x using a variation of an exact subgraph matching [5, 57] technique with backtracking capability in case a mapping failure occurs. The algorithm would then backtracks one step at a time and selects an alternative substrate nodes/links and repeats the mapping process again. The proposed algorithm is able to backtrack up to the VN root node. If within the selected h-level, none of the subnetworks were able to map the G^x, VM_h reports its failure to the SP, and the SP forwards the G^x one level up (i.e. $h-1$) in the hierarchy and the mapping process is repeated as shown in (Algorithm 1 Lines: 24-28. The SP forwards the G^x up twice in the hierarchy before the request is rejected. The embedding steps are described in details in both (Algorithm 1) and (Algorithm 2).

Algorithm 1: The proposed VN Embedding Algorithm, VNMA

Input: The substrate graph $G(N, L, \boldsymbol{c}, \boldsymbol{b})$ and a VN request $G^x(N^x, L^x, c^x, b^x)$

Output: VN embedding solution represented by the mapping functions: $\mathbf{M}_n^x$, $\mathbf{M}_l^x$

1 **if** (G^x) **then**
2 $\quad\Theta^x \leftarrow$ a vertex cover set of N^x;
3 $\quad\boldsymbol{vn}^x \leftarrow \{vn_1^x, vn_2^x, \cdots\}$ *such that* vn_i^x *is a star with root* $n_i^x \in \Theta^x$;
4 $\quad$CurrentCost $\leftarrow \infty$;
5 $\quad Mapped^x \leftarrow$ false;
6 $\quad h \leftarrow \min\{\delta\frac{|N|}{|N|^x}, k\}$;
7 $\quad$*// search for several embedding solutions in parallel in each subnetwork*
8 $\quad$**foreach** subnetwork g_{sub}^h, $sub = 1, \cdots, h$ **do**
9 $\quad\quad$Initialize: M_n^x and $M_l^x \leftarrow \Phi$;
10 $\quad\quad$MapVNStars(g_{sub}^h,$\boldsymbol{vn}^x$,M_n^x, M_l^x, Mapped^x);
11 $\quad$**end**
12 $\quad$*//find the embedding with the minimum cost*
13 $\quad$**for** $sub \leftarrow 1$ **to** h **do**
14 $\quad\quad$**if** (Mapped^x) **then**
15 $\quad\quad\quad$**if** (CurrentCost$> C(M_n^x)+C(M_l^x)$) **then** `// $C(M_n^x)$,$C(M_l^x)$ are the costs of the embedded nodes and links, respectively`
16 $\quad\quad\quad\quad\mathbf{M}^{\text{nodes}}(N^x)$,$\mathbf{M}^{\text{links}}(L^x) \leftarrow M_n^x$,$M_l^x$;
17 $\quad\quad\quad\quad$CurrentCost$\leftarrow C(M_n^x) + C(M_l^x)$
18 $\quad\quad\quad$**end**
19 $\quad\quad$**end**
20 $\quad$**end**
21 $\quad$**if** (Mapped^x) **then**
22 $\quad\quad$**return** $\mathbf{M}^{\text{nodes}}(N^x)$,$\mathbf{M}^{\text{links}}(L^x)$;
23 $\quad$**else**
24 $\quad\quad$*// if no mapping is found then search in higher-level subnetworks*
25 $\quad\quad$**if** $(h > 1)$ **then**
26 $\quad\quad\quad h \leftarrow h - 1$;
27 $\quad\quad\quad$GoTo 8 ;
28 $\quad\quad$**end**
29 $\quad\quad$**return** Mapped^x;
30 $\quad$**end**
31 **end**

Mapping the First Root of a VN Star, n_1^x

Mapping the first n_1^x onto $n_s \subset N_{sub}^h$ is straight forward:

- The mapping of a VN request starts by first selecting the first virtual node n_1^x in Θ^x that represents the root node of the vn_1^x to be mapped first.
- The substrate node n_s that is selected to map n_1^x is based on the current network residual resources. Such that the selected n_s has the highest available cpu in the subnetwork, as well as has sufficient bandwidth and node degree, that can satisfy the n_1^x constraints, (Algorithm 2: Lines 6-8).

Matching the VN Star vn_1^x

Once the n_1^x has been mapped, the next step is to map the remaining leaves (i.e. virtual nodes and links) of the star subgraph, vn_1^x. The leaves and the associated virtual nodes are concurrently mapped on each subnetwork using a variation of the exact subgraph matching algorithm with backtracking techniques as discussed in Section 4.5.3.

Once the star subgraph is mapped, the n_1^x and vn_1^x are removed from Θ^x and vn^x, respectively. Next, the VNMA will select the next star subgraph to be mapped from vn^x. Each star-subgraph in vn^x is individually mapped one at a time, the details of choosing the next vn_i^x is discussed below.

Algorithm 2: MapVNStars

Input: A subnetwork g^h_{sub}, a VN request stored as stars in $\boldsymbol{vn}^x$, the mappings, thus far, $\mathbf{M}^x_n$ and $\mathbf{M}^x_l$.

Output: updated mappings $\mathbf{M}^x_n$ and $\mathbf{M}^x_l$

if ($\boldsymbol{vn}^x$=ϕ) **then** `// all stars successfully mapped`
  Mapped$^x \leftarrow true$;
  return $\mathbf{M}^x_n$ and $\mathbf{M}^x_l$;
else
  Let n^x_i be the root of $vn^x_i \in \boldsymbol{vn}^x$ with highest degree and mapped neighbors;
  if ($M^x_n(n^x_i)$=ϕ) **then** `// the first` n^x_i `of` N^x `is not mapped yet`
    while ($\exists n_s \in g^h_{sub}$ satisfying CPU and degree constraints of n^x_i and Mapped$^x = false$) **do**
      $\mathbf{M}^x_n(n^x_i) = n_s$;
      Construct $G^{(\zeta)}(n_s)$;
      if (subGraphMatchingof($G^{(\zeta)}(n_s)$,vn^x_i)is successful) **then**
        Store results in $\{M^x_l(vn^x_i), M^x_n(vn^x_i)\}$;
        return MapVNStars(g^h_{sub}, $\boldsymbol{vn}^x - vn^x_i$, $M^x_n \cup M^x_n(vn^x_i)$, $M^x_l \cup M^x_l(vn^x_i)$, Mapped^x);
      end
    end
  else
    Let $n_s = M^x_n(n^x_i)$;
    Construct $G^{(\zeta)}(n_s)$;
    if (subGraphMatchingof($G^{(\zeta)}(n_s)$,vn^x_i)is successful) **then**
      Store results in $\{M^x_l(vn^x_i), M^x_n(vn^x_i)\}$;
      return MapVNStars(g^h_{sub},$\boldsymbol{vn}^x - vn^x_i$,$M^x_n \cup M^x_n(vn^x_i)$, $M^x_l \cup M^x_l(vn^x_i)$, Mapped^x);
    end
  end
  if (Mapped$^x = false$) **then** `// the mapping has failed`
    return $M^x_n = \Phi$ and $M^x_l = \Phi$;
  end
end

4.5.3 Modifying the VN Mapping to an Exact Matching Problem

The exact subgraph matching algorithm [5, 57] only performs a one-to-one matching between the nodes and links of any two graphs, as will be discussed in Section 4.4.1,hence it cannot be directly applied to the VN embedding problem. Although strict one-to-one mapping guarantees minimum cost, however, it implies a significant lower VN acceptance ratio when the network is congested.

To overcome this limitation, the neighboring area of the substrate node $n_s \in g^h_{sub}$ that is already been matched to the star root n^x_i is dynamically explored as follows: A new star metagraph is constructed, $G^{(\zeta)}_{sub}(n_s) \equiv (N^{(\zeta)}_{sub}, L^{(\zeta)}_{sub}, \boldsymbol{c}^{(\zeta)}_{sub}, \boldsymbol{b}^{(\zeta)}_{sub})$, as shown in (Algorithm 2), where ζ is the maximum allowable mapped path length (i.e. number of hops) as determined by the SP, such that

$$\begin{aligned} N^{(\zeta)}_{sub} &= \{n_s\} \cup \{\ n_d | n_d \in N^h_{sub}\}, \\ L^{(\zeta)}_{sub} &= \{l(n_s, n_d) | \forall p(n_s, n_d) \in P^h_{sub}, h(p) = j, j = 1, \cdots, \zeta\} \end{aligned}$$

where $h(p)$ is the number of hops in path $p(n_s, n_d)$. $\boldsymbol{c}^{(\zeta)}_{sub}$ is the vector of the residual capacities of the nodes participating in $G^{(\zeta)}(n_s)$. $\boldsymbol{b}^{(\zeta)}_{sub}$ represents the residual bandwidth of the links participating in $G^{(\zeta)}(n_s)$, such that $b^{(\zeta)}_{sub}(n_s, n_d)$ of a link $l(n_s, n_d) \in L^{(\zeta)}_{sub}$ corresponding to a path $p(n_s, n_d)$ equal to $\min\{R(l) : l \in p(n_s, n_d)\}$. Clearly, $G^{(\zeta)}_{sub}(n_s)$ is a star that can have multiple edges between any two nodes. These edges represent all paths between n_s and other nodes as far as ζ hops.

Fig. 4.4 shows an example of the constructed $G^{(\zeta)}_{sub}(n_s)$, where $\zeta = 2$, $n_s = F$. Node F has been previously selected to map the root of the star subgraph. In order to map the rest of the star, VNMA explores the underlying available network resources within the allowable ζ. The constructed graph is a star with node F as its root node as shown in Fig. 4.4b. Based on the required constraints of the star subgraph, the remaining leaves are mapped on $G^{(2)}(F)$ using exact graph matching with backtracking.

Once $G^{(\zeta)}_{sub}(n_s)$ is constructed, the scheme directly applies a commonly used exact subgraph matching heuristic [57] to match vn^x_i onto $G^{(\zeta)}_{sub}(n_s)$. The matching is conducted as follows: **First**, the substrate nodes that are previously mapped by the same VN request, or have insufficient resources are pruned from $G^{(\zeta)}_n$ to reduce the search time. **Second**, a number of candidate substrate nodes in $N^{(\zeta)}_{sub}$ are selected, from which, one node will

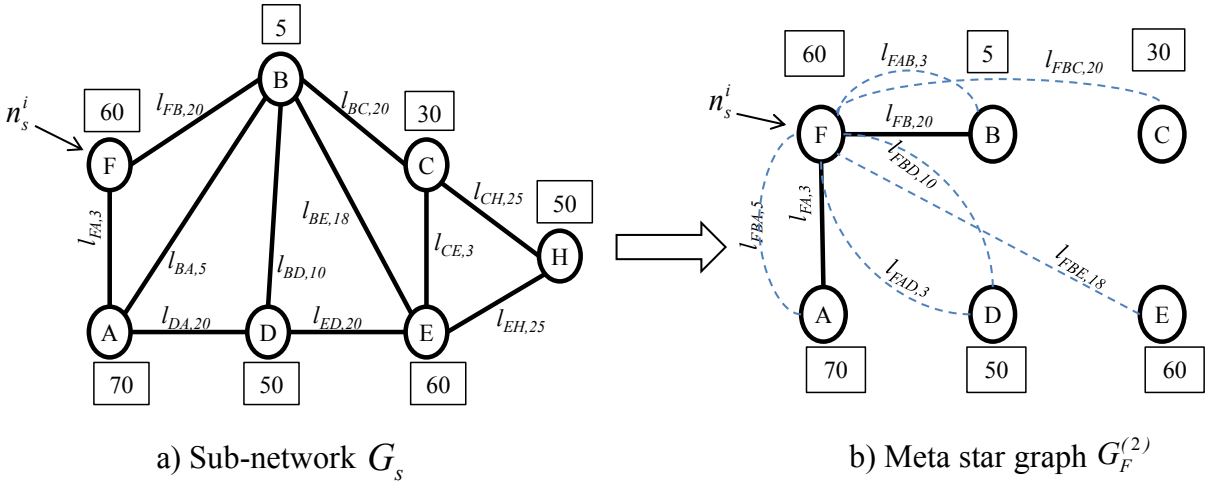

Figure 4.4: An example of the constructed meta subgraph $G^{(2)}$.

be chosen to map one of the unmapped virtual nodes in the star vn_i^x and its associated virtual link. The selection criteria for the candidate substrate nodes are based on the following sequence of priorities:

- A list is created that sorts the substrate nodes in $N_{sub}^{(\zeta)}$ based on their available cpu in a decreasing order.
- Traversing the list in order, the candidate substrate node that has the shortest path to the root node and has the highest available bandwidth across its path, i.e. $R(l_{sd}) \geq \boldsymbol{b}(l_{ij}^x)$ is selected and to map one of the virtual nodes and its associated virtual link.

Third, the mapped node and its path are temporary reserved for the vn_i^x and pruned from $G_{sub}^{(\zeta)}(n_s)$. Each path in $G_{sub}^{(\zeta)}(n_s)$ sharing a substrate link with the reserved path is updated with the temporary current available resources to avoid multiple assignments to the same link. **Fourth**, the algorithm proceeds with mapping the remaining branches of the star-subgraph. Once the algorithm maps the whole star-subgraph, it proceeds to the next vn_i^x, (Algorithm 2: Lines 17-20). During the course of mapping, if the algorithm cannot map the current star-subgraph, the scheme is capable of backtracking and mapping the previously mapped branches. The proposed algorithm can also be backtracked to the previously mapped star-subgraph and search for a different combination of substrate nodes and links, and then attempts mapping the current star-subgraph again. In

the worst case, backtracking can reach up to the first potential substrate root node in case no match is found. The scheme selects the new substrate node with the next highest available resources to be the new potential substrate root node and the whole mapping process is repeated again. **Finally**, upon the completion of the VN request mapping, s-each subnetwork reports its results (i.e. success or failure) to its VM_h-level coordinator with the associated mapping cost.

4.5.4 Selecting the Maximum Allowed Number of Hops, ζ

The selection of the maximum mapped virtual path length, ζ, enables the VN requests, that would otherwise be rejected, to be accepted. This leads to higher mapping costs and affects the future VN requests acceptance ratio as the network gets saturated. The experimental results have shown that setting ζ=2 or 3 achieves a balance between flexible mapping and limiting the VN mapping costs.

4.6 Performance Evaluation

In this section, first a description of the simulation setup for the proposed VNMA algorithm is discussed. Next, a comparison between the proposed hierarchical scheme versus flat network is conducted as one of the evaluation. Two different sets of VN requests are used in this simulation, the first set uses a small to medium sized VN requests (i.e. number of nodes), while the other set uses medium to big sized VN requests. The effect of varying δ is also discussed in this section. The simulation results are presented below and a detailed discussion of the performance follows. The performance metrics include, the effects of VN requests' acceptance ratio, the SP profit, the link and node utilization.

4.6.1 Experimental Settings

The GT-ITM tool [64] is used to generate all the random network topologies (i.e. the SN and the VN requests). The generated SN is setup to have 50-nodes. The nodes are randomly connected with a probability of 0.5 (i.e. around 600-links). The values of the substrate nodes' cpu and the substrate links' bandwidth are uniformly distributed

between 50 and 100. The hierarchical substrate network structure is created with $k = 4$ levels.

There are two sets of VN requests that are generated. In the first set, the number of generated virtual nodes is uniformly distributed between 2 and 10. In the second set, a larger size of VN requests are generated using a uniform distribution, the generated virtual nodes are between 5 and 14. The virtual links in both sets are randomly connected with a probability 0.5. The virtual nodes constraints (i.e. cpu) and the virtual links constraints (i.e. bandwidths) are also uniformly distributed between 0 and 20 and 0 and 50, respectively. The VN requests arrive following a Poisson process with an average of 4 requests per time window, and with infinite lifetime.

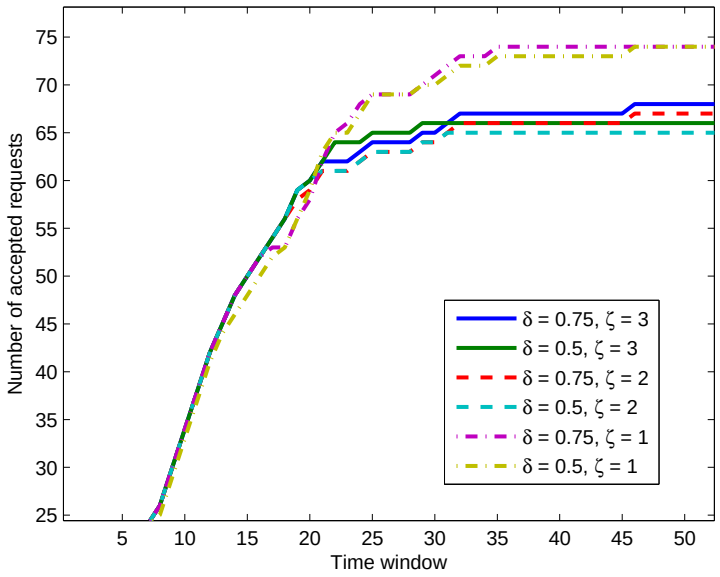

Figure 4.5: Number of accepted VN requests with variable δ.

4.6.2 Parametric h-level mapping selection

The effect of varying $\delta \in \{0.5, 0.75\}$ is shown in Fig. 4.5, Fig. 4.6 and Fig. 4.7. In this simulation, the set of a smaller sized VN requests is used for the evaluation. It is seen from Fig. 4.5 that, for a given δ, having larger δ may lead to a higher number of accepted requests. Higher δ also yields to higher profits as shown in Fig. 4.7. This

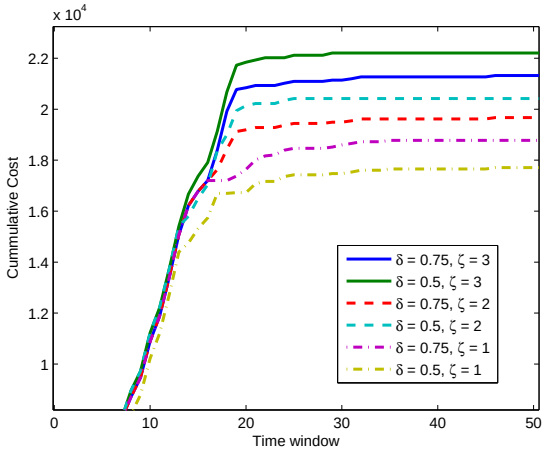

Figure 4.6: Cost with variable δ.

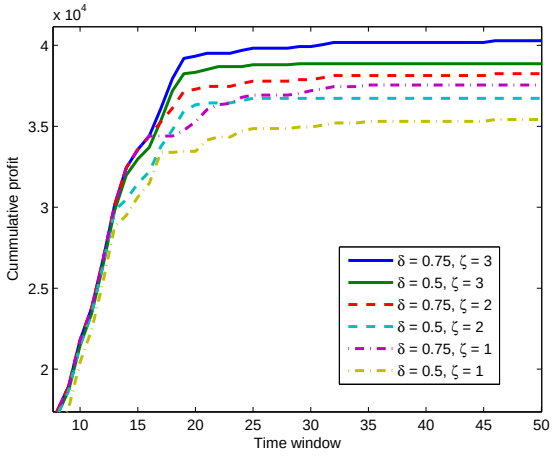

Figure 4.7: Profit with variable δ.

interesting behavior can be explained as follows. The choice of larger δ leads to selecting a larger h value in the hierarchy, as discussed in (4.8). The larger the h value is, the lower the position of the h^{th}-level coordinator in the hierarchy that is in charge of mapping the VN request. Mapping the VN request within the lower level of the hierarchy forces the mapped request to be within a closed neighborhood, that will in turn minimizes the mapped cost as shown in Fig. 4.6, and leads to a higher probability of having a more optimum result from the cost function in (4.6) and hence a higher profit.

4.6.3 Hierarchical versus flat substrate management frameworks

The performance of the proposed hierarchical VNMA mapping is also compared against the flat network. In this simulation, the second set of larger VN requests (i.e. 5 to 14) is employed to accentuate the advantages of the proposed hierarchical mapping framework. In this experiment, δ is set to 0.75, and $\zeta \in \{1, 2, 3\}$. Fig.4.8 shows that both schemes within the same ζ have similar acceptance ratio, however, in Fig. 4.10, the hierarchical mapping scheme has a higher profits. The hierarchical scheme has succeeded to minimize its cost function (4.6) by reducing its search space. The results of this evaluation underlines the potential of the efficient network resource allocation that this scheme provides.

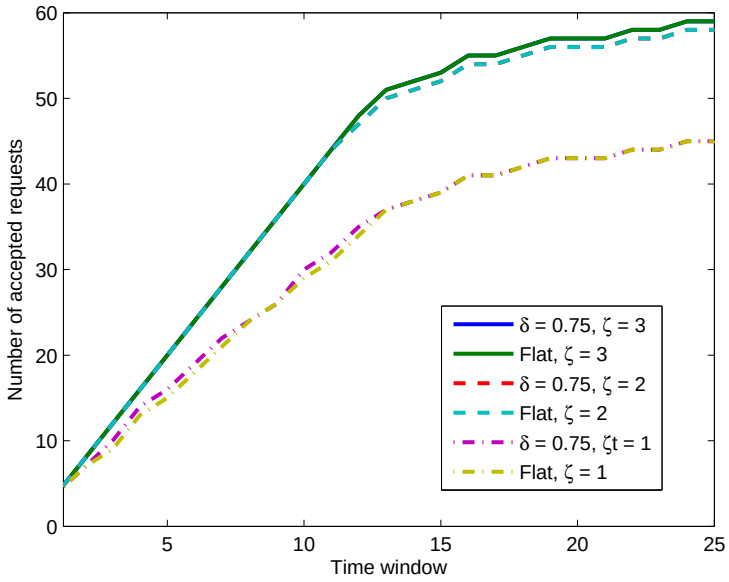

Figure 4.8: Effect of hierarchical versus flat frameworks on the number of accepted VNs.

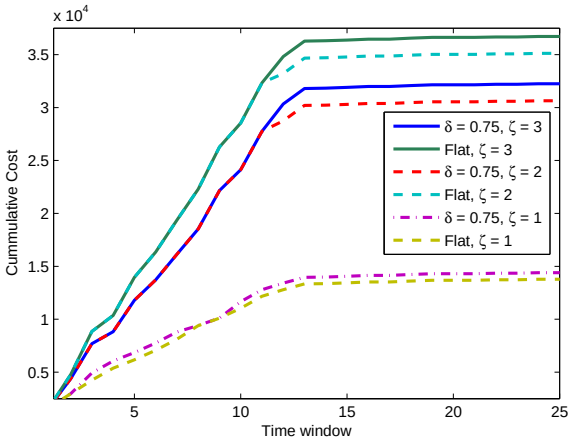

Figure 4.9: Effect of hierarchical versus flat frameworks on the cumulative cost.

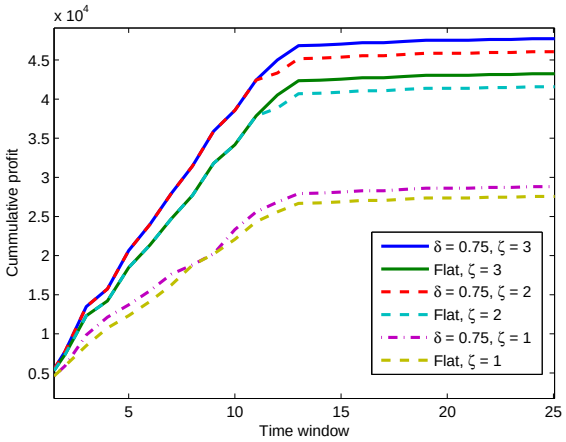

Figure 4.10: Effect of hierarchical versus flat frameworks on the cumulative profit.

4.6.4 Flexible number of hops (ζ)

The effects of having flexible number of hops (ζ) has also been tested. As shown in Fig. 4.5 and Fig. 4.7, the number of accepted requests and the associated profits follow different trends depending on the specified ζ. Despite the higher VN acceptance reported in Fig. 4.5 for $\zeta = 1$, it is evident from Fig. 4.7, however, that it corresponds to the least profit. This clearly shows that the number of accepted VN request does not necessarily convey the correct trend of the profits gained. The stringent constraint in finding the exact one-to-one mapping lead to rejecting the large VN requests with higher net profits while accepting much smaller requests with lower profits. In Fig. 4.8 and 4.10, the flexibility of choosing $\zeta > 1$ has shown to yield to better performance results by allowing the VNMA to continue to accept larger requests with higher profits. This also lead to better resource utilization while sustaining a balance-load across the network.

4.7 Summary

In this chapter, I have presented a novel hierarchical architecture for substrate management in order to realize efficient yet scalable virtual network (VN) mapping onto s substrate network (SN). The presented hierarchy concurrently finds more than one efficient solution for each VN request, hence increasing the chances of finding an optimal solution. Using this management hierarchy, a novel VN mapping scheme based on exact subgraph matching was described. Experimental results have shown the efficiency of the proposed scheme in terms of increasing the VN acceptance ratio and the network profit.

Chapter 5

Pricing Utility-Based VNs for Efficient Embedding onto a Shared SN

5.1 Introduction

The majority of previous research approaches focused mainly on the static VN embedding schemes assuming fixed resource requirements (e.g., CPU and bandwidth) throughout the VNs lifetime. However, practically traffic demands fluctuate due to the nature of the VN application (e.g. IPTV), network failure, etc.

With the wide adoption of network virtualization, both in academia through the use of VNs as efficient testbeds and in commercial applications (e.g., VNs hosting content delivery applications, IPTV [42, 61], on-demand gaming applications [26], etc.), such a static resource allocation may quickly deplete the SN resources. Moreover, this assumption does not take into consideration the wide spectra of the applications hosted on these VNs. That is, a number of the VN applications (e.g., testbeds) may not necessarily be sensitive to the time of execution, and can reduce resource consumption during periods of peak-demand. On the contrary, other VNs (e.g., those VNs hosting streaming applications) may require higher amounts of resources during certain periods, but substantially less resources during other periods.

The static resource allocation can be inefficient in practice. The VN users are charged for the reserved resources, irrespective of the actual use of these resources. With static resource allocation, the SP might deny the request of an embedded VN to temporarily increase its dedicated resources even if there is sufficient available resources, thus the SP is under utilizing its resources and hurting its revenues. A static VN economically hurts both the SP and the users.

This chapter presents a new pricing mechanism for VNs embedded onto a shared SN that is based on time-of-use. The SN hosts a variety of VNs each has its specific service or application. These VNs might not require fixed resources during all its lifetime, and they might be able to voluntarily adjust their required resource based on time to maximize their utility. Pricing resources is such an important economic incentive for the SN, and leads to better underlying network management [13].

Based on the traffic demand, the SP prices its resources to maximize its revenue and efficiently utilize its underlying resources. Through a proposed utility function that will be discussed later in this chapter, each VN establishes an optimal pricing scheme that is based on some budget constraints that maximizes its utility. Each VN requests a certain amount of physical resources that might be variable based on time-of-use.

In this chapter, we introduce two new terms; the *time-of-use* and the *VN scaling*. Both of these terms are studied as a mean for pricing policy of the underlying resources, and modeled their effects on the network utilization. The SP first presents its pricing scheme based on time-of-use to its new customers (i.e. VN requests) before mapping. Next, the preferences of the VN users are represented through a corresponding demand-utility function. The demand-utility function is based on the VN's hosted application sensitivity towards resource consumption and prices.

Performance evaluation of the proposed new pricing dynamic VN embedding scheme is presented and discussed at the end of this chapter to demonstrate the improvement achieved compared to the static VN embedding in terms of the resource utilization, revenues and the ratio of accepted VN requests.

5.2 Chapter Organization

The remainder of this chapter is organized as follows:

- Section 5.3 formulates the challenge of having fixed reserved physical resources throughout the VN lifetime. In the section, some definition that will be used throughout this chapter is also introduced.
- Section 5.4 discusses some of the pre-existing approaches related to the proposed work. The discussion will include different approaches towards resource pricing, congestion control, and dynamic management scheme of the allocated physical resources.
- Section 5.5 represents a brief description of the proposed dynamic system architecture
- Section5.6 discusses the details and the mathematical formulation of the proposed scheme.
- In Section 5.7 We show the simulation results and the evaluation of the proposed work.
- Finally, Section 5.8 concludes this chapter.

5.3 Problem Formulation and Definitions

The majority of the aforementioned approaches focus on the VN cost reduction or load balancing while assuming that the VNs resource consumption is static during all its lifetime, and that all VNs are regarded with equal priorities regardless of the traffic they carry and the sensitivity of their applications.

The SP achieves higher revenues, by efficiently utilizing its underlying resources while not sacrificing its guaranteed Quality of Service (QoS) to its customers (i.e. VNs). The InP leases its resources to SPs that host and accommodate a variety of VNs that perform different network applications and acquire different resource constraints. These network applications might be as simple as file transmitting or as complex as voice and video interaction transmission. Thus the expected QoS differ from one VN to the other, as well as it might not be consistent depending on the time-of-use. For example, the required streaming applications, like IPTV, might increase during the evening time and

might be reduced during the morning time. Thus if the reserved resources are static, then the resources are being under utilized during the less demand period, and the VNs are charged irrespectively of the actual use of resources. As well as with static resources, a network application might not be able to acquire a temporary higher resources than the actual reserved one, even if the SP has sufficient resources. The static resource allocation economically hurts both the service provider, i.e. SP, and the end-users, i.e. VNs.

Based on the above premise, it is clear that, the static resource allocation model although simple, yet not sufficient to reflect the demands of various VNs. The ability to differentiate between these VNs, and periodically reassess the optimality of the assigned resources is critical to the efficient utilization of the finite SN resources.

In this proposed work, we claim that the differentiation between various types of VNs is critical to achieving efficient substrate resource utilization and maximizing SPs revenues. The presented work also relates to the large body of research efforts related to pricing network resources (e.g., see [31, 41, 45]). To the best of the author's knowledge there has been no prior work for pricing VNs.

5.3.1 Definitions

In order to make the following discussion clearer, some important notations are defined below:

Definition 1. *The* **utility function***, as defined by the microeconomics, is the satisfiable level of resources allocated to users. In this work, the users satisfaction level is merely a combination of how much network resources are dedicated to him and their price based on time-of-use. Tailored to each user's network application, the user sets its utility function that maximizes the received resources while minimizing its cost in each specified time period (i.e. peak-time, mid-peak and off-peak). The utility function is attached to the VN request at the embedding time.*

Definition 2. *Based on the supply and demand in each time period, the SN markets its resources using a pricing scheme that is based on the* **time-of-use***. Time-of-use allows the SP to better utilize its network resources, and maximizes its profits. The time-of-use is diving into a specified number of periods (e.g peak and off-peak periods). During on-peak period (i.e. high demand), the SN prices its resources higher than the off-peak*

period to regulate resource consumption.

Definition 3. *In the* **VN scaling**, *the allocated physical resources (i.e. cpu and bandwidth) for a specific VN application might be scaled up or down based on demand. The VNs are only allowed to re-scale in the beginning of each new time period. The SP allows the VN scaling in order to efficiently utilize its underlying physical resources and maximize its revenues.*

5.4 Related Work

In this section, we present some of the previous work attempted to achieve guaranteed QoS to the end-users and at the same time to maximize the SP's profit from the economical point of view. The efficient resource allocation in the static network virtualization has been the focus of many researchers as the network resources has shown to be depleted quickly and are under-utilized in the constrained environment.

Fixed resource pricing by the SP has shown to be very inefficient, wastes resources and increase users' cost [15]. The authors in [15, 30, 41, 50, 65] argued that involving one pricing method or another has improved the network resource allocation over time and in turn has increased the revenues. The rest of this section discusses the relationship of pricing and traffic management.

5.4.1 Pricing and QoS

Network pricing in general has been extensively studied using variable resource pricing scheme. The objective is to study its impact on the users (i.e. increase/decrease number of users) and the offered QoS. Providing a guaranteed QoS requires the network to avoid congestion from occurring. Zhou et al. [65] assumed that the total required bandwidth by the VNs exceeds the available substrate network capacity. The authors focused on how to fairly allocate the underlying physical resources among its VNs using a non-cooperative game model and proposed a **pricing scheme** model to motivate the VNs to efficiently utilize the underlying resources.

Sayenko et al. [50] proposed a resource sharing model. The authors' objective was to

increase the service provider's revenue by grouping different network services into service classes with similar QoS requirement and allocating unused resources to the more expensive service class to increase the service provider's revenue.

Ibrahim et al. [30] proposed a resource management (RM) policy that is based on on a competitive market model. The authors argue that treating customers (i.e. users) equally will lead to under-utilized resources as well as unsatisfied customers seeking higher Qos Satisfaction Level (QSL). The authors proposed a new charging and RM scheme that is based on the amount of resources customers consume and their preferred QoS. Customers are charged differently based on the type of consumed resources (i.e higher or lower QoS).

5.4.2 Pricing and Congestion Control

Congestion in network occurs when the demand for resources exceeds the available capacity (i.e. cpu and bandwidth). Congestion may affect the expected packet delay, packets loss or even network failure. The dynamic pricing of network resources forces users to adapt to their resource usage, thus it is considered an affective tool to alleviate congestion [41]. As the network gets congested, network prices increase just discouraging users with tight budget to keep their sending rate intact. Users with tight budget adapt to the high prices by adjusting their sending rate. When the network is lightly loaded, network prices lower down just encouraging users to acquire more resources.

Odlysko [46] proposed the **Paris Metro Pricing (PMP)**, a simple network flat-rate pricing scheme to provide congestion control. PMP divides the network into several equal logically separated channels. All the channels treat the packets equally, however, each channel charges different price. The main idea is that channels that charge higher prices expect to be less congested, and thus will offer better service.

5.4.3 Pricing and Admission Control

Admission Control (AC) is used to control the load on the network by restricting some users from accessing the network in order to guarantee the desired QoS to the existing users, while efficiently utilize the network resources. The authors in [36] investigated the effect of pricing of the call admission in a wireless network environment, where a

dynamic pricing scheme is proposed that provides negative incentives according to the current network conditions as a mean to control congestion. In [22] an economic pricing tool is used to reflect demand and supply, such that, it encourages users with tight budget to postpone their calls at the times when the network is lightly loaded.

5.4.4 Pricing and Dynamic Resource Allocation

He et al. [27] were concerned about the poor utilization of the underlying resources that may lead to instability. The authors proposed **adaptive bandwidth allocation scheme** aided by the optimization theory leading to stability.

The authors in [16] proposed a fair variable pricing model for both the users and the service delivery by defining two functions, cost function and user utility function. Through this model, the users select the appropriate range of bandwidth in advance and its corresponding pricing to reduce its cost. The selected range of bandwidth allows the network to scale up or down the allocated bandwidth based on the available resources. In their work, the service delivery efficiently utilize its network resources and accept more users.

5.5 Pricing Utility-based VNs Embedding Architecture

In this chapter, a new pricing mechanism to provide efficient incentives for delay-tolerant VNs is proposed. In this work, the delay-tolerant VNs customize their resource consumption based on time-of-use. Consequently, a novel adaptive VN embedding scheme is developed that suits the new time-varying VN requests. To this end, the idea presented in this chapter can be summarized as follows:

- A new time-of-use pricing mechanism for the delay-tolerant network applications (i.e. VNs) is presented. The proposed pricing scheme is not directly coupled with the admission and congestion control, however, as will be discussed in the simulation in Section 5.7, through monitoring the SN utilization, this proposed work show better network utilization and an increase in the SP's revenues. In this work, different set of users (e.g. delay-tolerant and throughput sensitive) are investigated.

- To model the price sensitivity of the VN requests, a new dynamic VN model is presented that employs a utility function to describe the VN resource demands. This dynamic VN model will give users with tight budget the right economic incentive and is a reflection to the current network load. The SP advertises its network charges for each time period before the VN is embedded. The user then submits its required resources in each time period throughout its lifetime prior to embedding. Accordingly, the novel concept of *VN scaling* is introduced, where the VN dedicated resources are scaled up or down to continuously maximize the VN owners' utilities and minimize the embedding cost.

5.5.1 General System Architecture

The proposed architecture in this chapter focuses on providing optimal dynamic resource allocation between different VN classes based on time-of-use pricing scheme. The general system architecture mainly focuses on two main players: The SP and the end-user (i.e. VN). These two entities have different roles, characteristics and behavior in the VN environment are analyzed and discussed in Chapter 2.

However, unlike Chapter 2, the presented work has added extra attributes and characteristics to the VN. The VNs are divided into two service classes:

- Business Class VNs
  The business class VNs require a fixed network resources during all its lifetime. This specific class of VNs are price and congestion level indifference (i.e. The VN owners are willing to pay more during the peak/mid-peak times to guarantee fixed resources).
- Economy Class VNs
  The VNs with economy service class are users with tight budget and flexible resource requirement over time, with the objective of maximizing the value obtained from the SN resource consumption. The economy class VNs are willing to tolerate a reduction (i.e. down-scale) in their reserved resources during peak/mid-peak times or at high congestion level to reduce their cost. During the off-peak time and low congestion level the economy class VNs acquire higher (i.e. up-scale) resources.

The time in this proposed work is divided into three main periods: on-peak, mid-peak

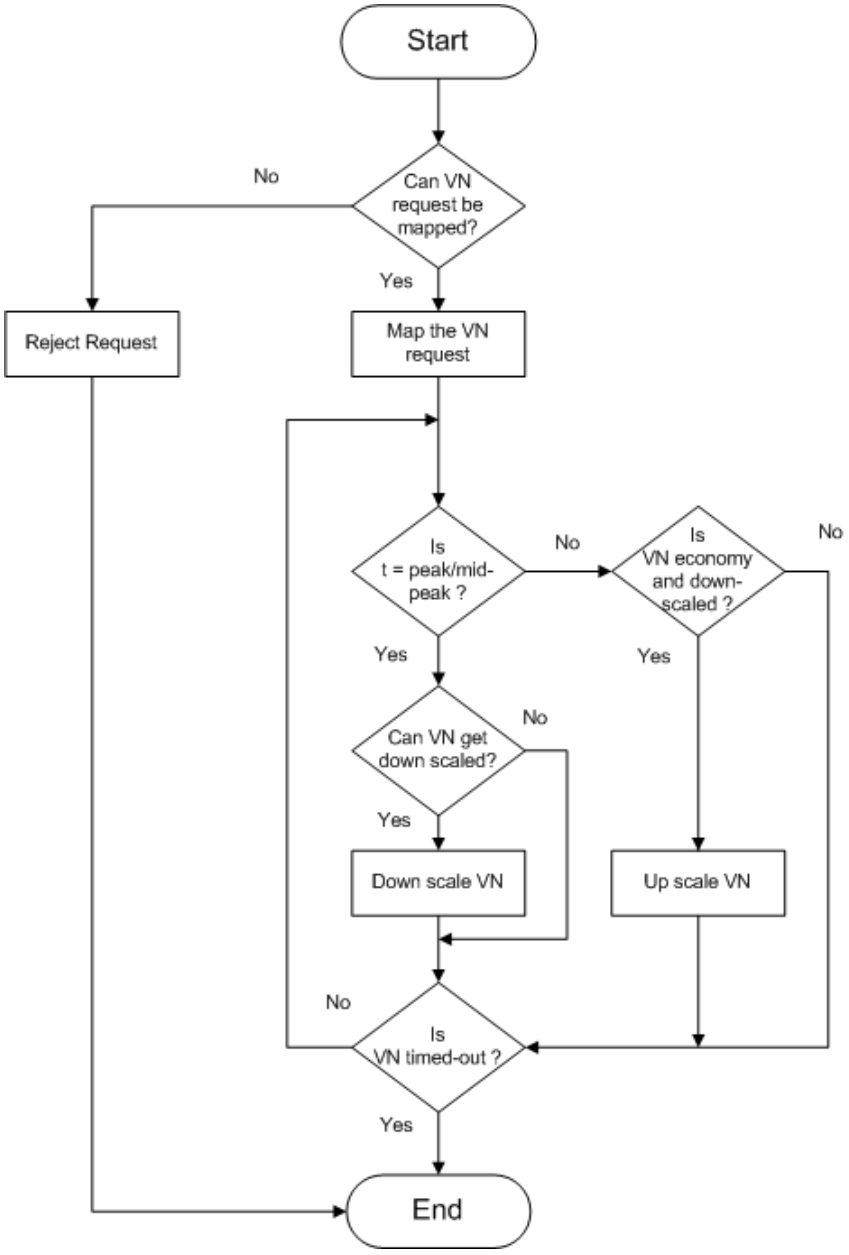

Figure 5.1: General System Architecture.

and off-peak periods. The on-peak period represents the time when the network is highly loaded and has high congestion level. High congestion level might effect packets loss or undesired packet delay. In the off-peak period the network is lightly loaded and does not suffer from any congestion. On the other hand, the mid-peak period represents the transition between the on-peak period and off-peak period.

The SP sets a different price scheme for each time period, with the on-peak period having the highest price and the off-peak period having the lowest price. The SP advertises its prices to the new end-users before their requested VNs get embedded. In order for the users to maximize their utility, each user based on its network application and its willingness to pay, determine its demanded resources for each time period, see Section 5.6.1 for more details. Each end-user sends its resource request for each time period to the SP. Based on the requested resources, the SP runs its mapping scheme to map the VN request.

When a VN request in the SP queue is ready to be mapped at time t , the SP first performs admission control to guarantee sufficient physical resources to accommodate the requested VN. If sufficient resources are available, the SP maps the VN request using the proposed VNMA algorithm, as discussed in Chapter 4.5, regardless of its service class. Once the VN is mapped the proposed algorithm checks the current time period, the characteristics of the mapped VN and the resource demand for each time period. If the current time period is peak/mid-peak and the VN belongs to the economy class and it is a good candidate for down scaling, the VN is down-scaled as shown in Fig. 5.1 to reduce its cost. In order to reduce unnecessary network overhead, the SP only selects from the economy class the VNs with high expected lifetime and high revenues to be allowed scaling.

Based on the characteristics of each time period, the SP adjusts the VNs resource demand accordingly. The SP performs up-scaling/down-scaling only at the beginning of each time period to reduce the excessive network overhead. The proposed pricing scheme allows the SP to accepts more VN requests and efficiently utilize its resources. Once the VN expires, the SP releases back the reserved resources, and makes them available for the future VNs.

5.6 Utility-based Dynamic VN Embedding Scheme

Since the underlying resources are shared and limited, the aforementioned VN static model discussed in Chapter 4 may result in a quick depletion and/or defragmentation of these resources. Furthermore, it does not provide an efficient economic incentive for delay-tolerant VNs to regulate their resource consumption. In fact, previous approaches (see [11, 18]) have shown that, at a relatively high rejection ratio for the VN requests, the SN is usually highly under-utilized.

In this chapter, in order to efficiently utilize the underlying resources and overcome the aforementioned limitations, a novel dynamic utility-based VN model is introduced. Using this model, I will show how pricing and periodic reconfiguration, for the delay-tolerant users, through up- and down- scaling operations solves the under-utilized resources problem and maximize the VN objective utility function in a QoS-enabled network environment.

5.6.1 Proposed Dynamic Utility-Based VN model

In contrast to the existing static VNs model, see Chapter 4, the proposed model allows delay-tolerant VN users to modify their required resource dynamically. The SP allows resource modifications to occur only at the beginning of each time period, with the objective of maximizing the value obtained from the physical resources. The SP's operating time cycle (e.g., one day) is divided into Q periods indexed by t, where $t = 1, 2, \cdots, Q$, with an equal period length $\triangle T$. The SP prices its resources differently based on t. The operating life cycle indicates the time interval taken into account for decision making in VN scheduling. The choice of this interval depends on the behavior of the VN users and their demand patterns.

Next, the notion of dynamic VNs by using G_t^x to denote a VN request with time-dependent resource allocation is expressed, where t denotes a given time period. In this model, G_t^x is obtained by adapting G^x through up- or down-scaling the requested resources in the following manner:

$$G_t^x \equiv G^x \left(N^x, L^x, r_t^x \mathbf{c}^x, r_t^x \mathbf{b}^x\right) \tag{5.1}$$

where $r_t^x \geq 0$ is a t-dependent scalar multiplier that either up- or down-scales the re-

quested virtual nodes' cpu and the virtual links' bandwidth of the economy class VN request G^x at time period t. Note that, the $r_t^x = 1$ for the business class VN requests.

The resource demands of a VN user x vary according to the criticality and tolerance of the application running on the VN. Each VN owner determines the optimal amount of resources represented by the optimal values for the scaling multipliers, grouped in a vector $r^x := \left(r_1^x, r_2^x, \cdots, r_Q^x\right)$, that will, from the VN owner's perspective, maximizes its gain. The preferences of each VN owner for the required resource usage is modeled by a *utility function*, $U^x\left(G^x, r^x\right)$, that quantifies the value obtained in terms of the amount of allocated resources $r_t\mathbf{c}^x$ and $r_t^x\mathbf{b}^x$ at time t. When users do not consume any VN resources they gain no benefit, hence, $U^x\left(G^x, \mathbf{0}\right) = 0$.

The VN users are charged according to their reserved physical resources at each period t. For this purpose, the SP maintains a price vector $\rho = (\rho_1, \cdots, \rho_Q)$, such that $\rho_t = (\rho_t^{\text{cpu}}, \rho_t^{\text{bw}})$, t=1, $\cdots$, Q, where ρ_t^{cpu} and ρ_t^{bw} are the prices of a unit of substrate node processing cycles and link bandwidth, respectively, at time t. To encourage users to shift some of their demands to off-peak periods, the SP dynamically adjusts its prices to reflect the demand on each specific period, such that, the on-peak period will have the highest price. After the SP announces its ρ, each new user finds the optimal values for r^x that will accommodate its network application over the whole time cycle. The mathematical derivation of the optimal r^x is carried out in the next subsection.

Derivations of the optimal scaling vector r^x

From the VN owner's perspective, the optimal r^x is the one that maximizes its utility $U^x\left(G^x, \boldsymbol{r}^x\right)$ while minimizing the price paid to the SP, which is given by $\sum_{t=1}^{Q} r_t^x \boldsymbol{d}^x.\rho_t'$, where the prime ($'$) on a vector indicates its transpose, and $\boldsymbol{d}^x$=$(d_{\text{cpu}}^x, d_{\text{bw}}^x)$ represents the vector of the VN request resource demands, per unit time, such that:

$$d_{\text{cpu}}^x = \sum_{n_i^x \in N^x} c_i^x \quad , d_{\text{bw}}^x = \sum_{l_{ij}^x \in L^x} b_{ij}^x \tag{5.2}$$

In the proposed dynamic VN model, the user can choose to up- or down-scale the needed VN resources during the Q periods through the vector $\boldsymbol{r}^x$ such that the net amount of consumed resources in the static and dynamic models does not change, i.e.,

$\sum_{t=1}^{Q} \triangle T r_t^x \boldsymbol{d}^x \leq \tau_x \boldsymbol{d}^x$. Mathematically, the user's problem can be formulated as follows:

$$\mathbf{P1}: \underset{\boldsymbol{r}^x}{\text{maximize}} \quad \left(U^x\left(G^x, \boldsymbol{r}^x\right) - \sum_{t=1}^{Q} r_t^x \boldsymbol{d}^x.\rho_t' \right) \tag{5.3}$$

$$\text{s.t.} \quad \sum_{t=1}^{Q} \triangle T r_t^x \leq \tau_x \tag{5.4}$$

In addition to the nonnegativity constraints $r_t^x \geq 0$, $\forall t$. Here, a user x is only concerned with the resource allocation problem during a single SP operation cycle.

The problem **P1** can be solved by first formulating the Lagrangian [40] $\mathcal{L}(\boldsymbol{r}^x, \lambda_x)$,

$$\mathcal{L}\left(\boldsymbol{r}^x, \lambda_x\right) \quad \equiv \quad U^x\left(G^x, \boldsymbol{r}^x\right) \quad - \quad \sum_{t=1}^{Q} r_t^x \boldsymbol{d}^x.\rho_t' \quad + \quad \lambda_x \left(\frac{\tau_x}{\triangle T} - \sum_{t=1}^{Q} r_t^x \right) \tag{5.5}$$

where $\lambda_x \geq 0$ is the Lagrangian multiplier. The optimal values of r^x and λ_x that maximize (5.5) follow from the Karush-Kuhn-Tucker (KKT) conditions [40]:

$$\begin{cases} \frac{\partial U^x(G^x, \boldsymbol{r}^x)}{\partial r_t^x} - \boldsymbol{d}^x.\rho_t' = \lambda_x & \forall r_t^x > 0 \\ \frac{\partial U^x(G^x, \boldsymbol{r}^x)}{\partial r_t^x} - \boldsymbol{d}^x.\rho_t' < \lambda_x & \forall r_t^x = 0 \end{cases} \tag{5.6}$$

$$\frac{\tau_x}{\triangle T} - \sum_{t=1}^{Q} r_t^x = 0. \tag{5.7}$$

Thus, given the price vector ρ, candidate optimal values [1] for r^x and λ_x are determined through solving the system (5.6) - (5.7). Since $U\left(G^x, \boldsymbol{r}^x\right)$ is concave and the constraint (5.4) is linear, the obtained solution also maximizes **P1** [40]. An optimal r^x can either be a corner solution with $r_t^x = 0$ for some t or an interior one with $r_t^x > 0$, $\forall t$. When an interior solution exists, the right-side of (5.6) can be used to construct a system of $Q-1$ independent equations in $\boldsymbol{r}^x$ of the form:

$$\frac{\partial U^x(G^x, \boldsymbol{r}^x)}{\partial r_t^x} - \boldsymbol{d}^x.\rho_t' = \frac{\partial U^x(G^x, \boldsymbol{r}^x)}{\partial r_{t'}^x} - \boldsymbol{d}^x.\rho_{t'}' \tag{5.8}$$

These $Q-1$ equations along with the constraint (5.7) can then be solved to obtain the optimal vector $\boldsymbol{r}^x$. The following section gives some examples of utilities for VNs hosting different types of applications.

[1] A unique solution is guaranteed if the user utility is strictly concave.

Examples of $U^x(G^x, r^x)$

In the following examples, an operating cycle with a peak and an off-peak periods ($t = 1, 2$, respectively) is considered, where for simplicity, $U^x(.)$ is only dependent on $\boldsymbol{r}^x$.

- **Utilities with perfect substitution of periods.** One example of these utilities are those that represent delay-tolerant VN requests where hosted applications of the VNs can adjust their resource consumption according to the price elasticity. These VNs can for example host a testbed dedicated to the execution of several independent experiments [51]. Those VN requests can be represented by the utility function $U^x\left(G^x, \boldsymbol{r^x}\right) = \alpha r_1^x + \beta r_2^x$. It can be easily shown that the optimal scaling for G^x for a price $\boldsymbol{\rho}$ is:

$$(r_1^x, r_2^x) = \begin{cases} (\frac{\tau_x}{\triangle T}, 0) & \frac{\alpha - \boldsymbol{d}^x.\rho_1'}{\beta - \boldsymbol{d}^x.\rho_2'} > 1 \\ (0, \frac{\tau_x}{\triangle T}) & \frac{\alpha - \boldsymbol{d}^x.\rho_1'}{\beta - \boldsymbol{d}^x.\rho_2'} < 1 \\ (\frac{\sigma\tau_x}{\triangle T}, \frac{(1-\sigma)\tau_x}{\triangle T}), & \sigma \in [0, 1], \text{Otherwise} \end{cases}$$

  The above solution indicates that if $(\alpha - \beta) > \boldsymbol{d}^x.(\rho_1 - \rho_2)'$, then the only demand for the VN will be in the peak-time and vice versa. On the other hand, when $(\alpha - \beta) = \boldsymbol{d}^x.(\rho_1 - \rho_2)'$ the utility of resources in both periods becomes equal, and any $\boldsymbol{r}^x$, that satisfies the constraint in (5.7), is optimal.

- **Elastic utilities.** Here the VN user is willing to switch between the resources at the two periods at a fixed ratio based on the price. For instance, the utility can take the form $U^x(G^x, \boldsymbol{r}^x) = \alpha \log r_1^x + \beta \log r_2^x$, where Log utilities are usually employed to model demands of multimedia applications such as on-demand gaming [49]. Using (5.6), the optimal $\boldsymbol{r}^x$ can be found that satisfies, in addition to the constraint in (5.7), the relation: $\frac{\alpha}{r_1^x} - \boldsymbol{d}^x.\rho_1' = \frac{\beta}{r_2^x} - \boldsymbol{d}^x.\rho_2'$.

- **Utilities with perfect complement periods.** Another example of VN demand-utility is when the VN owner would like to maintain a fixed ratio between the usage of one period and the other. For example, the utility $U^x(G^x, \boldsymbol{r}^x) = \min\{r_1^x, \alpha r_2^x\}$ reflects the user's desire to consume exactly $\alpha > 0$ as much in the on-peak period as in the off-peak one. One example of such utility is a testbed experimentation that requires a period of high resource utilization for the execution of the experiment followed by the use of less resources for the processing and analysis of the results. Given any price vector, the VN owner will always choose $\boldsymbol{r}^x$ that satisfies $\frac{r_1^x}{r_2^x} = \alpha$ and

the constraint (5.7). In other words, the net utility can increase only by increasing the scaling multipliers in both periods.

5.6.2 Pricing Virtual Networks

The previous section addressed the VN subscriber's problem of finding the optimal r^x given a price ρ. This section considers the problem of maximizing the *net social welfare* of the system comprised of the VN users and the SP [49]. In this system, a socially fair allocation is the one that maximizes all the VN users' utilities while minimizing the SP's total cost. I first note that each VN user individually selects its scaling vector r^x that maximizes its own utility, given a certain price vector, by solving **P1**. As, the user's decision does not consider the SP cost or the utilities of other users. Hence, mathematically, the problem of maximizing the social welfare of the system can be formulated as follows.

$$\begin{aligned} \mathbf{P2}: \underset{\boldsymbol{r}^1,\cdots,\boldsymbol{r}^{\mathcal{X}}}{\text{maximize}} & \sum_{x=1}^{|\mathcal{X}|} \left(U^x(G^x, \boldsymbol{r}^x) - C(G^x) \right) \\ \text{s.t.} & \sum_{t=1}^{Q} r_t^x . \triangle T \leq \tau_x, \qquad x = 1 \cdots, |\mathcal{X}| \end{aligned} \tag{5.9}$$

As in the static VN model, $C(G^x)$ is the cost of embedding the request of user x during the time in which the VN request is active. This cost is analyzed in the following two sections for the case of dynamic VNs.

Formulation of the VN embedding cost, $C(G^x)$

The embedding cost $C(G^x)$ in the dynamic model represents the sum of the operating cost of the CPU allocated in the substrate nodes, i.e. $M_t^{\text{nodes}}(N^x)$, and the bandwidth assigned in the substrate links, i.e. $M_t^{\text{links}}(L^x)$, at each time period, with the subscript t added to the mapping functions to indicate the embedding period t. More precisely,

$C(G^x)$ for the dynamic utility-based VNs is obtained as follows:

$$C(G^x) = \sum_{t=1}^{Q} r_t^x \triangle T \left(\omega_{\text{cpu}} \sum_{n_i^x \in N^x} c_i^x + \omega_{\text{bw}} \times \right. \\ \left. \sum_{l_{ij}^x \in L^x} |p_{kt}(\mathbf{M}_t^{\text{nodes}}(n_i^x), \mathbf{M}_t^{\text{nodes}}(n_j^x))| b_{ij}^x \right) \quad (5.10)$$

Recall from 4.3.2, that ω_{cpu} and ω_{bw} are constant factors that depend on the operational costs of the SP resources. Here, $p_{kt}(\mathbf{M}_t^{\text{nodes}}(n_i^x), \mathbf{M}_t^{\text{nodes}}(n_j^x))$ is the unique path that maps a link l_{ij}^x at t. (5.10) can be rewritten as

$$\begin{aligned} C(G^x) &= \overbrace{\tau_x \boldsymbol{d}^x . \boldsymbol{\omega}'}^{\text{fixed cost}} \\ &+ \omega_{\text{bw}} \triangle T \sum_{t=1}^{Q} r_t^x \sum_{l_{ij}^x \in L^x} \Big(|p_{kt}(\mathbf{M}_t^{\text{nodes}}(n_i^x), \\ &\quad \mathbf{M}_t^{\text{nodes}}(n_j^x))| - 1 \Big) b_{ij}^x \end{aligned} \quad (5.11)$$

where $\boldsymbol{\omega}' = (\omega_{\text{cpu}}, \omega_{\text{bw}})'$. From (5.11), $C(G^x)$ has a fixed component that is independent of the VN owner's scaling vector, r^x, and a component that is increasing in both the vector of scalers r^x and the length of the substrate paths with more than one link, i.e., for $|p_{kt}| > 1$, $p_{kt} \in M_t^{\text{links}}(L^x)$.

While this cost is incurred by the SP, VN users are only charged for the amount of the requested resources $\sum_{t=1}^{Q} r_t^x \boldsymbol{d}^x . \rho_t'$ and not for any additional resources reserved for the VN. Hence, to maximize its profit, the SP aims at minimizing the cost of the extra allocated resources, as indicated by the second component in (5.11), that is used to satisfy the VN requests. If an SP can map each VN link onto one single substrate link (i.e. one-to-one mapping), then the second component in the cost function vanishes. On the other hand, the longer the paths that service VN links are, the higher the incurred cost is.

The objective of finding shortest path links is inherently embedded in the functionality of all VN embedding schemes. However, as the physical resources become scarce, the embedding scheme is forced to employ longer paths to service the VN requests. Hence, the second component in (5.11) can be thought of as the congestion cost resulting from VNs competing over the physical resources. By estimating the VN hosting cost, the SP

can derive an efficient pricing scheme ρ that propagates these costs to the VN users. The users, in turn, can adjust their resource demands to consume less resources during peak-demand periods, hence, decreasing the congestion cost. Unfortunately, the actual values for $|p_{kt}|$ cannot be known a-priori and can only be calculated after the VN embedding scheme is executed. One simple way to overcome this problem is to actively monitor the SN at each period to obtain an average path length $\hat{h}_t$. $\hat{h}_t$ can then be employed in (5.11) to provide the SP with an estimate for the expected cost at each period, which can be written as:

$$C(G^x) \simeq \tau_x \boldsymbol{d}^x.\boldsymbol{\omega'} + \omega_{\text{bw}} d^x_{\text{bw}} \triangle T \sum_{t=1}^{Q} r^x_t (\hat{h}_t - 1) \tag{5.12}$$

While simple, the above approach is coarse and can not accommodate slight variations in the users' VN requests.

Dynamically adjusting the SN prices

The SP dynamically adjusts its pricing scheme using the concept of a responsive pricing mechanism [19]. Responsive pricing is a mean for congestion control during peak time. Responsive pricing best suits users with adaptive resources. The SP uses historical measurements of its resource utilization at each time period to set its price $\boldsymbol{\rho}$, and the delay tolerant VN users adjust their resources accordingly.

The SP posts its price $\boldsymbol{\rho}$ to its new users to make their decisions by solving **P1**. Then, using the new load of the SN, the SP recomputes and posts a new price vector at the beginning of the new operating cycle and so on. During on-peak period when the network is highly congested, the responsive pricing increases the resource pricing. Users with tight budget and delay tolerant tend to either reduce their reserved resources or shift their resources during off peak time, when the SP decreases its prices.

Once the end-users receive $\boldsymbol{\rho}$, they determine their optimal $\boldsymbol{r}^x$ and submit their requests to the SP to perform the embedding step which is described in the following section. By switching some of the traffic to the off-peak period, the SP manages to efficiently utilize its network resources at any specific period, and hence maximize its profits.

5.6.3 The Time-based VN Embedding Scheme

As indicated in the pervious section, the SP collects measurements about the SN utilization in each period t, and then posts a price $\boldsymbol{\rho}$ to the new VN users. These users calculate their optimal $\boldsymbol{r}^x$ and submit their requests for each time period to the SP to be embedded.

A received request G_t^x is stored in a service queue, dequeued at the beginning of each time t by the SP, as discussed in Chapter 4, and sent to the appropriate VM_h to be embedded. If the VN was inactive at $t-1$ (i.e., $r_{t-1}^x = 0$) it is embedded as a new request. At the beginning of each time period (i.e. on-peak, mid-peak or off-peak) each VM_h checks its already active VNs at $t-1$ for re-scaling up or down according to their r_{t-1}^x and r_t^x based on time-of-use. The VN embedding steps is described in details in Algorithm 3 and discussed in Chapter 4.

5.6.4 Embedding a Re-scaled VN request

When the SP receives a request to down-scale a specific VN (i.e., $r_t^x < r_{t-1}^x$), VM_h that has embedded the request is notified to simply reduce the amount of allocated resources to the already existing VN and then reports back the incurred cost (5.12). The VM_h meanwhile initiates a new search in its h subnetworks for another optimum embedding. The objective is with downscaling the pre-mapped VN, searching for another match might result in an optimum match that will further reduces the SP cost as shown in (5.12).

Finally, if a better embedding is found, VM_h re-embeds the VN accordingly. Similarly, when the SP receives a request to up-scale a VN (i.e., $r_t^x > r_{t-1}^x$), the corresponding VM_h attempts to increase the allocated resources to the VN. If it succeeds it reports back the cost, otherwise it reports a failure, and a new search to embed the VN is initiated by the SP. The proposed algorithm periodically re-partitions the SN to allow the SP to efficiently utilize the underlying resources, hence, reducing the defragmentation of these resources.

Algorithm 3: The proposed VN Embedding Algorithm at period t

Input: The substrate graph $G(N, L, \boldsymbol{c}, \boldsymbol{b})$ and a VN request $G^x(N^x, L^x, c^x, b^x)$ with a usage scaling vector $\boldsymbol{r^x}$

Output: VN embedding solution represented by the mapping functions: $\mathbf{M}_n^x$, $\mathbf{M}_l^x$

1 **if** ($r_{t-1}^x = 0$) **then** // G^x *not active at* $t-1$
2 $\quad\Theta^x \leftarrow$ a vertex cover set of N^x;
3 $\quad\boldsymbol{vn}^x \leftarrow \{vn_1^x, vn_2^x, \cdots\}$ *such that* vn_i^x *is a star with root* $n_i^x \in \Theta^x$;
4 $\quad$CurrentCost $\leftarrow \infty$;
5 $\quad Mapped^x \leftarrow$ false;
6 $\quad h \leftarrow \min\{\delta\frac{|N|}{|N|^x}, k\}$;
7 $\quad$// *search for several embedding solutions in parallel in each subnetwork*
8 $\quad$**foreach** subnetwork g_{sub}^h, $sub = 1, \cdots, h$ **do**
9 $\quad\quad$Initialize: M_n^x and $M_l^x \leftarrow \Phi$;
10 $\quad\quad$MapVNStars(g_{sub}^h,$\boldsymbol{vn}^x$,M_n^x, M_l^x, Mapped^x);
11 $\quad$**end**
12 $\quad$//*find the embedding with the minimum cost*
13 $\quad$**for** $sub \leftarrow 1$ **to** h **do**
14 $\quad\quad$**if** (Mapped^x) **then**
15 $\quad\quad\quad$**if** (CurrentCost$> C(M_n^x)+C(M_l^x)$) **then** // $C(M_n^x)$,$C(M_l^x)$ *are the costs of the embedded nodes and links, respectively*
16 $\quad\quad\quad\quad\mathbf{M}^{\text{nodes}}(N^x)$,$\mathbf{M}^{\text{links}}(L^x) \leftarrow M_n^x$,$M_l^x$;
17 $\quad\quad\quad\quad$CurrentCost$\leftarrow C(M_n^x) + C(M_l^x)$
18 $\quad\quad\quad$**end**
19 $\quad\quad$**end**
20 $\quad$**end**
21 $\quad$**if** (Mapped^x) **then**
22 $\quad\quad$**return** $\mathbf{M}^{\text{nodes}}(N^x)$,$\mathbf{M}^{\text{links}}(L^x)$;
23 $\quad$**else**
24 $\quad\quad$// *if no mapping is found then search in higher-level subnetworks*
25 $\quad\quad$**if** ($h > 1$) **then**
26 $\quad\quad\quad h \leftarrow h - 1$;
27 $\quad\quad\quad$GoTo 8 ;
28 $\quad\quad$**end**
29 $\quad\quad$**return** Mapped^x;
30 $\quad$**end**
31 **else**
32 $\quad$Re-scale(G^x,r_{t-1}^x,r_t^x);
33 **end**

5.7 Performance Evaluation

In this section, the proposed dynamic utility-based VN model is evaluated and compared to the fixed VN model in terms of the VN acceptance ratio, resource utilization and revenues. The proposed mode is implemented using MATLAB. Furthermore, GT-ITM tool [64] was employed to generate the topologies of both the substrate network and the VN requests.

- **Time:**
  The simulation time unit is taken to represent an hour and the experiments are

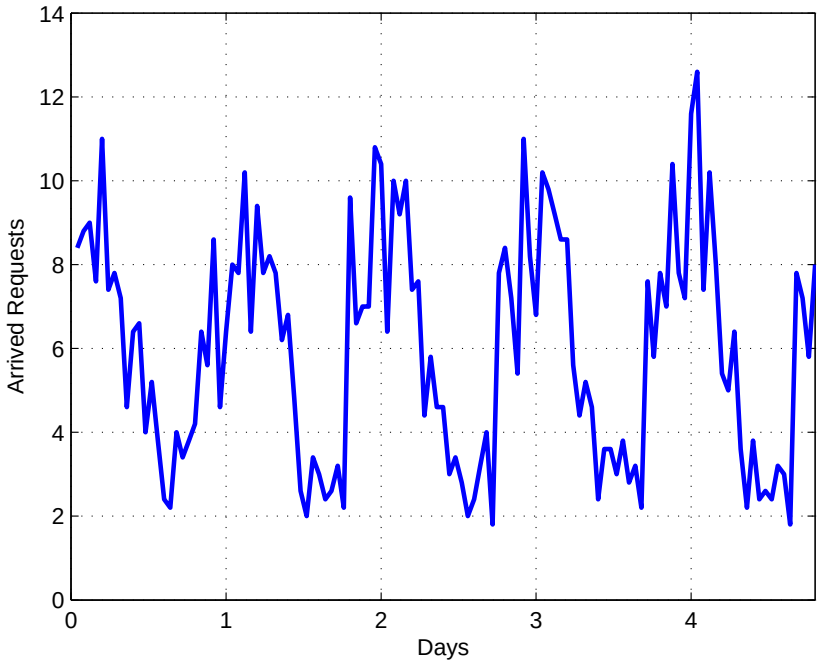

Figure 5.2: Arrived VN requests per time period.

  run for 5 days (120 time units). Each day is divided into three periods $Q = 3$, namely, off-peak, mid-peak and on-peak demand periods. Each period lasts for $\triangle T = 8$ time units as shown in Fig. 5.2.

- **The Substrate Network:**
  Similar to existing approaches [11, 17], the substrate network is generated using

	Economic	**Business**
Fixed	–	100%
Dynamic	20%	80%

Table 5.1: Traffic Scenarios.

50-nodes. The substrate nodes are randomly connected with a probability of 0.5 (i.e. around 600 substrate links). The nodes and the links capacities are both uniformly distributed between $50 - 100$MHz and $50 - 100$Mbps, respectively. The number of levels in the visualization management hierarchy K and the value for δ have been set experimentally to 4 and 0.75, respectively.

- **The Virtual Networks:**
  The size of the generated VN requests use uniform distribution that ranges between 2 to 10 nodes. Similar to the substrate network, the virtual nodes are randomly connected with a probability of 0.5. The virtual nodes' and links' capacities are both uniformly distributed between $5 - 20$MHz and $5 - 50$Mbps, respectively. The VNs have a lifetime that follows an exponential distribution with $\mu = 20$ hours. The arrived VNs are chosen such that the business class VNs represent 80% of the total arrived VN requests in the dynamic pricing mechanism, and 100% in the fixed pricing scheme, as shown in Table 5.1. The VN requests arrival at the off-peak, mid-peak and on-peak time periods follows a Poisson distribution with means 3, 6 and 9, requests per hour, respectively.

- **The VN Embedding and Scaling Scheme:**
  Both the VN dynamic pricing mechanism and the VN fixed pricing scheme use the proposed VNMA algorithm for the VN requests mapping mechanism. Both algorithms use $p_k = 3$ as the maximum allowable path length as discussed in Chapter 4. The dynamic utility-based approach selectively chooses from the economy VNs the ones with the high revenues and long lifetime as good candidates for scaling.

- **Simulation Approach:**
  In the following sections, the performance of the proposed embedding scheme using the VN dynamic pricing is evaluated against the VN fixed pricing mechanisms. Four different metrics were employed in this evaluations: the VN acceptance ratio (i.e., the ratio between the number of accepted requests and the total number of arrived

requests per time period), link/node utilization which reflects the ratio between the used bandwidth/CPU on each link/node to its total capacity, and finally, we also show the improvement in the total profit defined as the difference between the amount paid by the VN users and the SP cost of embedding the VNs.

5.7.1 VN Acceptance Ratio

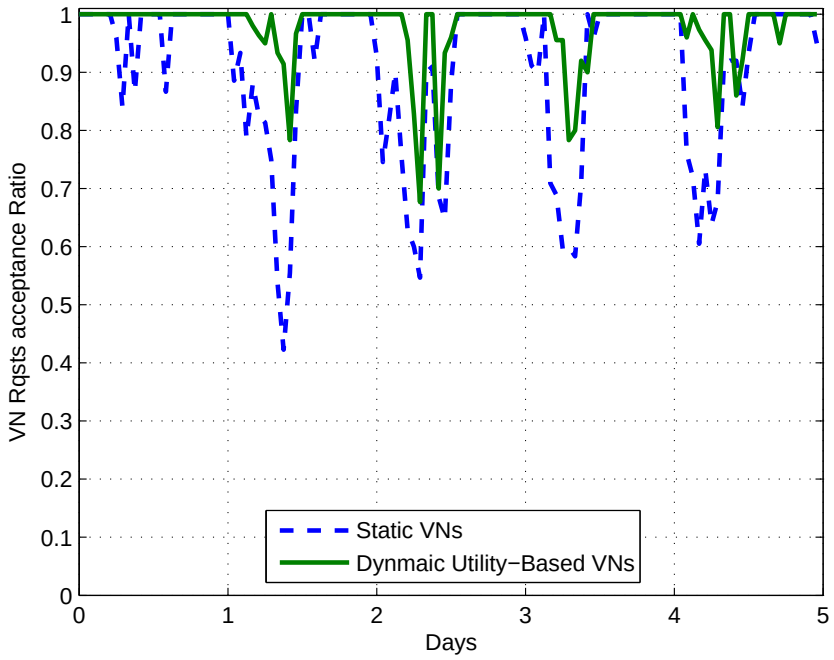

Figure 5.3: Acceptance ratio of the received VN requests.

Fig. 5.3 depicts the achieved VN acceptance ratio for the proposed embedding scheme using the dynamic utility-based pricing mechanism versus that with the fixed pricing, and as shown the utility-based dynamic VN model has better acceptance ratio. The average arrival VN acceptance ratio for both schemes is summarized in Table 5.2.

These results can be attributed to the fact that during the off-peak period, the network has a low arrival rate, and some of the previously mapped VN have already expired. Thus the network experiencing low demand and that's making the acceptance ratio similar. However, in the on-peak period the dynamic utility-based pricing scheme down-scales

Time period	**Utility-based dynamic VN model**	**Fixed VN model**
$off - peak, \lambda = 3$	99%	96%
$mid - peak, \lambda = 6$	93%	86%
$on - peak, \lambda = 9$	98%	83%

Table 5.2: VN Acceptance Ratio.

the selected economy VNs and re-optimize its mapping solution, thus allowing for more new VN requests to be accepted then in the fixed pricing scheme.

5.7.2 Utilization

The node utilization: The effect of having higher VN acceptance ratio in the dynamic

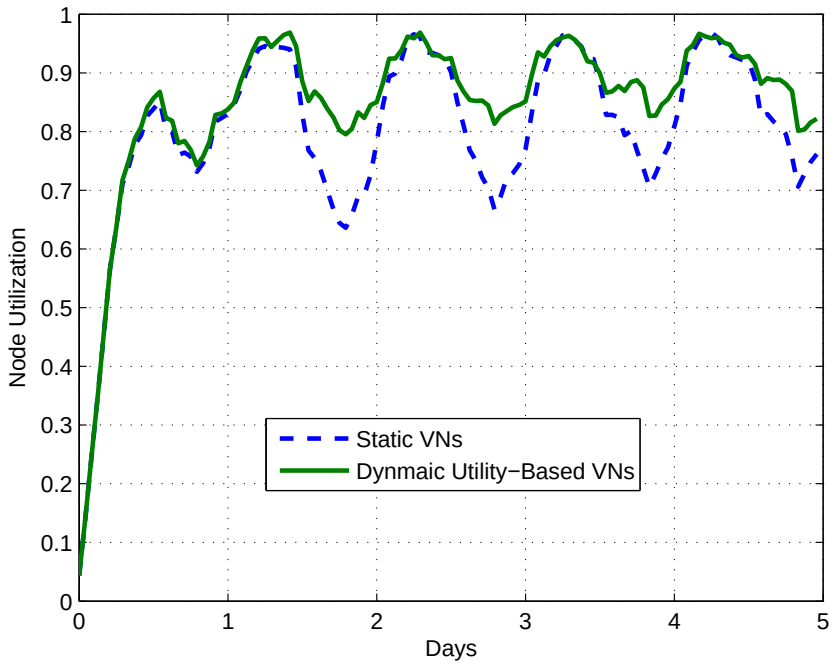

Figure 5.4: Average substrate node utilization.

utility-based during the on-peak period is not reflected in Fig. 5.4, that is because both schemes have reached the maximum node utilization. However, the dynamic utility-based

Time period	**Utility-based dynamic VN model**	**Fixed VN model**
$off - peak, \lambda = 3$	88%	82%
$mid - peak, \lambda = 6$	89%	83%
$on - peak, \lambda = 9$	92%	88%

Table 5.3: Node Utilization.

pricing scheme has shown to have a smoother curve, this is due to the up-scaling of the pre-mapped VNs during the off-peak time (i.e. the network is lightly loaded). Table 5.3 shows the average node utilization in both schemes for different time periods, where the dynamic utility-based VN model is shown to have better node utilization.

Fig. 5.4 indicates that around 11% of the CPU of the whole SN cannot be allocated to any VNs, this is due to:

- Resource fragmentation phenomenon, where the resources are scattered all over the SN.
- The node mapping is based on one-to-one mapping, such that, if the requested resources are more than the available substrate resources, then these substrate resources will not be fully utilized.

The link utilization: The link utilization as shown in Fig. 5.5 follow the same trend as that of the node utilization. The dynamic utility-based VN model scheme has shown to have better link utilization than of the fixed VN model scheme, see Table 5.4 This phenomenon can be explained as follows:

- At the on-peak periods the increase in $\boldsymbol{\rho}_t$ forces some pre-mapped VNs that are price sensitive to down-scale their resource consumption, hence, freeing more substrate resources.
- The proposed algorithm attempts to re-map the down-scaled VNs with the objectives of finding a shorter path, thus reducing its cost and freeing more resources.
- Performing periodic repartition for the underlying resources achieves load balancing and helps the embedding algorithm to efficiently utilize the underlying resources.

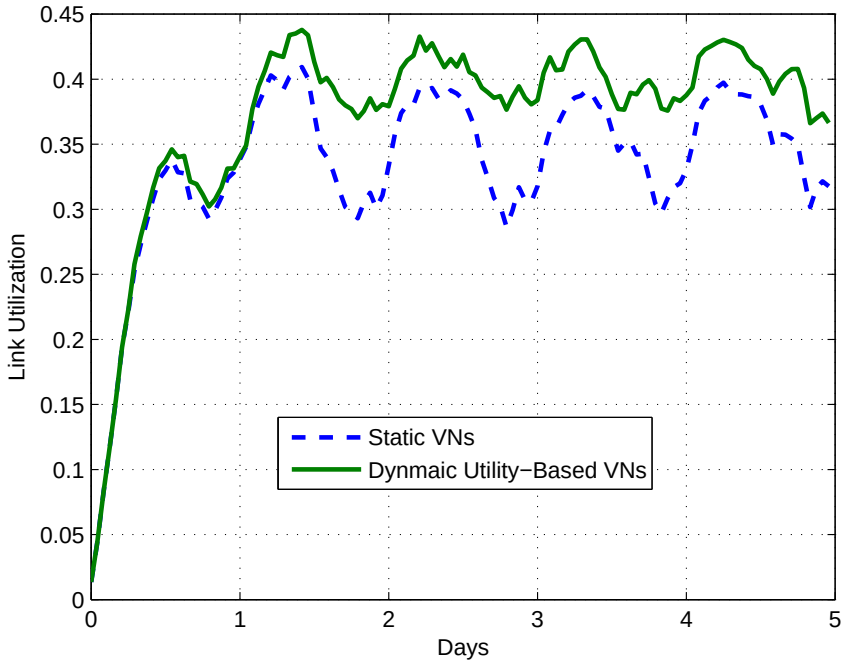

Figure 5.5: Average substrate link utilization.

Time period	**Utility-based dynamic VN model**	**Fixed VN model**
$off - peak, \lambda = 3$	39%	35%
$mid - peak, \lambda = 6$	40%	34%
$peak, \lambda = 9$	40%	36%

Table 5.4: Link Utilization.

- The dynamic utility-based dynamic scheme experienced lower congestion level during peak-time due to the re-scaling process compared to that of the fixed scheme, that further allowed the proposed scheme to accept more VN requests.

Re-scaling lead to higher node and link Utilization: The introduction of a dynamic pricing mechanism forces some pre-mapped VNs with tight budget to down-scale their reserved resource consumption during on-peak time, that in turn lead to a reduction in the node and link utilization; however, it has been offset by the SP accepting more VN requests leading to an overall higher link utilization as compared to that of the fixed pricing scheme. Even at off-peak periods, the dynamic utility-based VN model has much

better and smoother performance than of the static VN model. The reason is that, during the off-peak period, the substrate network has low network congestion and in turn lowers its price $\boldsymbol{\rho}_t$. Some pre-mapped economic class VNs opt to up-scale their resource usage, leading again to a higher node and link utilization.

The node utilization, as shown in Fig. 5.4, follows the same trend as that of the link utilization. This shows that the employed dynamic pricing enhances and smoothes the substrate network resource usage. However, it is noted that the gap between the two pricing mechanisms in Fig.5.4 is smaller than Fig.5.5, this can be explained as the congestion is mainly attributed to longer paths, leading to higher link utilization, unlike the node mapping that only allows one-to-one mapping.

5.7.3 Cost vs. Profit

The effect of having higher VN acceptance ratio during the on-peak period, and the option to upscale the reserved resources of the economy VN class during the off-peak period, have lead the dynamic utility-based VN embedding scheme to incur higher cost than in the VN fixed embedding scheme as shown in Fig. 5.6. However, adopting the dynamic utility-based VN embedding scheme has lead to higher profits as shown in Fig. 5.7. This profit is maintained during the off-peak period as a result of the economy class VN users shifting their requests to the off-peak period or acquiring more resources. Even during the on-peak period, the dynamic utility-based scheme has higher profits due to the business VN class that are willing to pay higher prices to keep its reserved resources fixed. Thus dynamic utility-based VN model scheme allowed the SP to efficiently utilize its resources and increase its profits

5.8 Conclusion

This chapter presented a novel dynamic pricing mechanism for virtual networks (VNs) embedded onto a shared substrate network. A time varying VN requests based on demand-utility functions is presented. Experimental results have shown that, the proposed dynamic VN embedding scheme in terms of the VN acceptance ratio, resource utilization and network profit have shown better results compared to the fixed VN em-

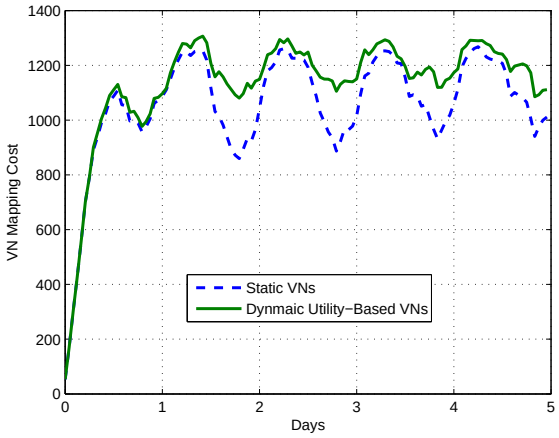

Figure 5.6: The SP cost per time period.

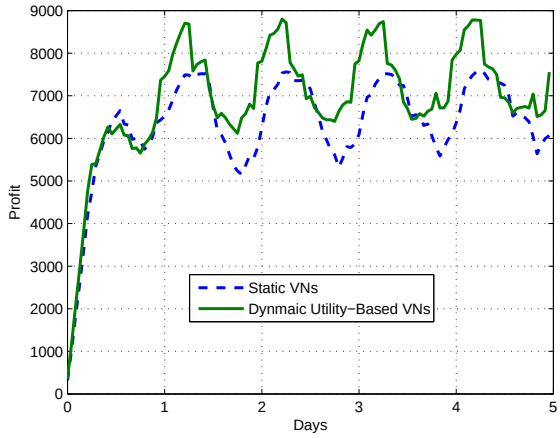

Figure 5.7: The SP profit per time period.

bedding scheme.

Chapter 6

Conclusions and Future Work

Network virtualization has been proposed as a gateway to overcome the limitation of the current Internet infrastructure by allowing VNs to share the underlying resources. An optimal resource discovery and allocation techniques are needed to efficiently utilize the limited physical resources.

In this thesis, we have addressed the problem of VN embedding problem onto a shared SN. We first considered the general problems facing the VN embedding schemes, and later we addressed some of the existing work that have been proposed.

A new hierarchical substrate management framework has been proposed that is based on k-level hierarchical virtualization management that forces the VN requests to be embedded within a closed neighborhood to minimize the incurred costs by the SP and efficiently utilize the underlying resources. Performance evaluation illustrated the efficiency of the proposed scheme, when the scheme is compared against the flat network.

Next, a new pricing mechanism for time-varying VNs is also proposed to regulate the physical demand of the substrate resources, to achieve better network utilization, and to avoid unnecessary congestion. Experimental results demonstrate the time-varying VNs achieved higher profits for SP, better resource utilization and higher VN acceptance ratio compared to the fixed VNs resources.

However, some improvements must be addressed in future work:

- In this proposed work, we have neglected the switching cost of re-scaling VNs due to migrating the virtual nodes and links which can result in service disruption to some VN applications (e.g., those hosting VoIP). To reduce this effect, we restricted such a process to the beginning of each period instead of allowing a continuous price adjustment at a finer time granularity. Nonetheless, this switching cost can also be further taken into consideration by modifying the utility functions of the VN requests to include the cost of service interruption.
- Another possible extension to the proposed work is to allow the VN users to express resource utilities at the level of individual VN links and nodes, thereby, allowing each user to request the re-scaling of certain components of the VN graph. We believe that a solution to this problem can follow from the same steps taken in the proposed work, where the scalers r_t^x are defined as vectors of scalers, one for each VN link or node.

Bibliography

[1] David G. Andersen. Theoretical approaches to node assignment, 2002.

[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34 – 41, april 2005.

[3] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Omni: An efficient overlay multicast infrastructure for real-time applications. *Comput. Netw.*, 50(6):826–841, April 2006.

[4] John Adrian Bondy. *Graph Theory With Applications*. Elsevier Science Ltd, 1976.

[5] Bunke, H. Recent Developments in Graph Matching. In *15th Int. Conf. on Pattern Recognition, Barcelona*, pages 813–817, 2000.

[6] Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente, and Daniel Villela. A survey of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 29(2):7–23, April 1999.

[7] Jorge Carapinha and Javier Jiménez. Network virtualization: a view from the bottom. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*.

[8] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *SIGCOMM Comput. Commun. Rev.*, 41(2):38–47, April 2011.

[9] M. Chowdhury, M.R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, 20(1):206 –219, feb. 2012.

[10] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 49–56, New York, NY, USA, 2010. ACM.

[11] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. pages 783 –791, april 2009.

[12] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *INT J PATTERN RECOGN*, 18(3):265–298, 2004.

[13] Luiz A. DaSilva. Pricing for qos-enabled networks: A survey. *Communications Surveys Tutorials, IEEE*, 3(2):2 –8, quarter 2000.

[14] Zhenhai Duan, Zhi li Zhang, and Yiwei Thomas Hou. Service overlay networks: Slas, qos and bandwidth provisioning. *IEEE/ACM Transactions on Networking*, 11:870–883, 2002.

[15] R. Edell and P. Varaiya. Providing internet access: what we learn from index. *Network, IEEE*, 13(5):18 –25, sep/oct 1999.

[16] Adel S. Elmaghraby, Anup Kumar, Mehmed M Kantardzic, and Mostafa Gamal Mostafa. Bandwidth allocation in a dynamic environment using a variable pricing policy. *Computers and Communications, IEEE Symposium on*, 0:589, 2002.

[17] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –6, june 2011.

[18] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. Vnr algorithm: A greedy approach for virtual networks reconfigurations. In *Global Telecommunications Conference (GLOBECOM 2011)*, pages 1 –6, dec. 2011.

[19] Matthias Falkner, Michael Devetsikiotis, and Ioannis Lambadaris. An overview of pricing concepts for broadband ip networks. *Communications Surveys Tutorials, IEEE*, 3(2):2 –13, quarter 2000.

[20] N. Farooq Butt, M. Chowdhury, and R. Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. *NETWORKING 2010*, pages 27–39, 2010.

[21] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, January 2007.

[22] D.F. Ferguson, C. Nikolaou, and Y. Yemini. An economy for flow control in computer networks. In *INFOCOM '89. Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies. Technology: Emerging or Converging, IEEE*, pages 110 –118 vol.1, apr 1989.

[23] N.C. Fernandes, M.D.D. Moreira, I.M. Moraes, L.H.G. Ferraz, R.S. Couto, H.E.T. Carvalho, M.E.M. Campista, L.H.M.K. Costa, and O.C.M.B. Duarte. Virtual networks: Isolation, performance, and trends. *Annals of Telecommunications*, pages 1–17, 2010.

[24] T. Ghazar and N. Samaan. Pricing utility-based virtual networks for efficient embedding onto a shared substrate network. *IEEE Transactions on Network and Service Management, (second round of revisions).*

[25] T. Ghazar and N. Samaan. Hierarchical approach for efficient virtual network embedding based on exact subgraph matching. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1 –6, dec. 2011.

[26] F. Hao, TV Lakshman, S. Mukherjee, and H. Song. Enhancing dynamic cloud-based services using network virtualization. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 37–44. ACM, 2009.

[27] J. He, R. Zhang-Shen, Y. Li, C.Y. Lee, J. Rexford, and M. Chiang. Davinci: Dynamically adaptive virtual networks for a customized internet. In *Proc. ACM CoNEXT*. ACM, 2008.

[28] I. Houidi, W. Louati, and D. Zeghlache. A distributed virtual network mapping algorithm. In *ICC'08*, pages 5634–5640, 2008.

[29] Ines Houidi, Wajdi Louati, Djamal Zeghlache, Panagiotis Papadimitriou, and Laurent Mathy. Adaptive virtual network provisioning. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 41–48, New York, NY, USA, 2010. ACM.

[30] W. Ibrahim, J.W. Chinneck, S. Periyalwar, and H. El-Sayed. Qos satisfaction based charging and resource management policy for next generation wireless networks.

In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 2, pages 868 – 873, june 2005.

[31] R. Johari and J.N. Tsitsiklis. A scalable network resource allocation mechanism with bounded efficiency loss. *Selected Areas in Communications, IEEE Journal on*, 24(5):992 – 999, may 2006.

[32] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Supercomputing, 1998. SC98. IEEE/ACM Conference on*, page 28, nov. 1998.

[33] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, December 1998.

[34] Eric Keller, Ruby B. Lee, and Jennifer Rexford. Accountability in hosted virtual networks.

[35] G. Kontesidou and K. Zarifis. Openflow virtual networking: A flow-based network virtualization architecture. `http://www1.icsi.berkeley.edu/~zarifis/pubs/FULLTEXT01.pdf/`.

[36] Tianshu Li, Youssef Iraqi, and Raouf Boutaba. Pricing and admission control for qos-enabled internet. *Comput. Netw.*, 46(1):87–110, September 2004.

[37] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proc. of the 1st ACM workshop VISA 2009*, pages 81–88. ACM, 2009.

[38] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. *Tech. Rep. WUCSE-2006-35, Washington University, USA, Tech. Rep*, 2006.

[39] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005.

[40] M. Minoux. *Mathematical Programming: Theory and Algorithms. Chichester*. U.K.: Wiley, 1986.

[41] J.K. MacKie-Mason and H.R. Varian. Pricing congestible network resources. *Selected Areas in Communications, IEEE Journal on*, 13(7):1141 –1149, sep 1995.

[42] C. Marquezan, L. Granville, G. Nunzi, and M. Brunner. Distributed autonomic resource management for network virtualization. In *IEEE/IFIP Network Operations and Management Symposium, 19-23 April 2010, Osaka, Japan*, pages 463–470. IEEE, 2010.

[43] C.C. Marquezan, J.C. Nobre, L.Z. Granville, G. Nunzi, D. Dudkowski, and M. Brunner. Distributed reallocation scheme for virtual network resources. In *ICC'09*, pages 1–5. IEEE, 2009.

[44] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):493 –504, may 1998.

[45] A. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE Trans. Smart Grid*, 1(3):320 –331, dec. 2010.

[46] A. Odlyzko. Paris metro pricing: the minimalist differentiated services solution. In *Quality of Service, 1999. IWQoS '99. 1999 Seventh International Workshop on*, pages 159 –161, 1999.

[47] Vasileios Pappas, Beichuan Zhang, Andreas Terzis, and Lixia Zhang. Fault-tolerant data delivery for multicast overlay networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*.

[48] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64, January 2003.

[49] M. Saad, A. Leon-Garcia, and Wei Yu. Optimal network rate allocation under end-to-end quality-of-service requirements. *Network and Service Management, IEEE Transactions on*, 4(3):40 –49, dec. 2007.

[50] A. Sayenko, T. Hamalainen, J. Joutsensalo, and J. Siltanen. On providing bandwidth and delay guarantees using the revenue criterion based adaptive wfq. In *Communications, 2003. APCC 2003. The 9th Asia-Pacific Conference on*, volume 2, pages 745 – 750 Vol.2, sept. 2003.

[51] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture: proposal and initial prototype. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, VISA '09, pages 63–72, New York, NY, USA, 2009. ACM.

[52] T. Stutzle and H. H. Hoos. Max-min ant system. *Future Gener.Comput. Syst.*, 16:889–914, 2000.

[53] W. Szeto, Y. Iraqi, and R. Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 6, pages 3004 – 3008 vol.6, dec. 2003.

[54] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. A multi-agent systems approach to autonomic computing. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*.

[55] Jonathan Turner. A proposed architecture for the geni backbone platform. In *In Proc. Architecture for Networking and Communications Systems*, pages 12–2006, 2006.

[56] vpn. Bgp/mpls rfc2547. `http://tools.ietf.org/html/rfc2547/`.

[57] G. Wang, B. Wang, X. Yang, and G. Yu. Efficiently indexing large sparse graphs for similarity search. *IEEE Trans. Knowl. Data Eng*, (99):1–1, 2009.

[58] Ju Wang. Tolerating denial-of-service attacks using overlay networks impact of overlay network topology. In *in 2003 ACM Workshop on Survivable and Self-Regenerative Systems. 2003. Washington DC: ACM*, 2003.

[59] M. Wang, C. W. Tan, W. Xu, and A. Tang. Cost of not splitting in routing: Characterization and estimation. *IEEE/ACM Trans. Networking*, PP(99):1, 2011.

[60] D. Warneke and O. Kao. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):985 –997, june 2011.

[61] H. Yin, X. Liu, G. Min, and C. Lin. Content delivery networks: a bridge between emerging applications and future ip networks. *Network, IEEE*, 24(4):52–56, 2010.

[62] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Comm. Rev.*, 38(2):17–29, 2008.

[63] Minlan Yu, J. Rexford, Xin Sun, Sanjay Rao, and N. Feamster. A survey of virtual lan usage in campus networks. *Communications Magazine, IEEE*, 49(7):98 –103, july 2011.

[64] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *INFOCOM'96*, volume 2, pages 594–602.

[65] Ye Zhou, Yong Li, Guang Sun, Depeng Jin, Li Su, and Lieguang Zeng. Game theory based bandwidth allocation scheme for network virtualization. In *Global Telecommunications Conference (GLOBECOM 2010)*, pages 1 –5, dec. 2010.

[66] Y. Zhu and M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In *INFOCOM 2006*, pages 1–12. IEEE, 2006.

[67] Yaping Zhu, Rui Zhang-Shen, Sampath Rangarajan, and Jennifer Rexford. Cabernet: connectivity architecture for better network services. In *Proceedings of the 2008 ACM CoNEXT Conference*.