A RESOURCE PUBLICATION AND DISCOVERY FRAMEWORK AND BROKER-BASED ARCHITECTURE FOR NETWORK VIRTUALIZATION ENVIRONMENT

Sleiman Rabah

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science (Software Engineering) at

Concordia University

Montréal, Québec, Canada

January 2014

© Sleiman Rabah, 2014

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sleiman Rabah**

Entitled: **A Resource Publication and Discovery Framework and Broker-Based Architecture for Network Virtualization Environment**

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

Master of Applied Science (Software Engineering)

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

Signed by the final examining committee:

_____ Chair
Dr. Yuhong Yan

_____ Examiner
Dr. Todd Eavis

_____ Examiner
Dr. Lingyu Wang

_____ Supervisors
Dr. Joey Paquet and Dr. Rachida Dssouli

Approved by _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____
Dr. Christopher Trueman, Interim Dean
Faculty of Engineering and Computer Science

Abstract

A Resource Publication and Discovery Framework and Broker-Based Architecture for Network Virtualization Environment

Sleiman Rabah

The Internet has received a phenomenal success over the past few decades. However, the increasing demands on the Internet usage and the rapid evolution of the applications and services provided over the Internet have demonstrated that the current Internet architecture is unsuitable for supporting many types of applications. Moreover, its ubiquity and multi-provider nature make nearly impossible the introduction of radical changes or improvements without coordination and consensus between many providers. Thus, any technological changes in the current Internet architecture could result in unintended consequences on the overall Internet usage. Network virtualization is considered as promising, yet challenging, solution to overcome these limitations. It commonly refers to the creation of several isolated logical networks that can coexist on the same shared physical network infrastructures. Its key concept is to enable several network architectures to run concurrently in a multi-role-oriented environment in which the role of the traditional Internet Service Provider (ISP) is decoupled into several roles such as infrastructure provider (InP), virtual network provider (VNP) and service provider (SP). Despite the promising benefits, this concept is associated with many challenges. These, among others, include the description and publication as well as discovery of resources on which virtual networks are deployed.

In this thesis, we define a broker-based architecture that provides functions for publishing, discovering and negotiating as well as instantiating and managing resources in network virtualization environment. We proposed an information model that assists various providers in describing the resources and services they offer and we implemented a proof of concept

prototype to demonstrate the feasibility of the proposed architecture. Moreover, we have conducted extensive experiments to evaluate the performance and the scalability of the implemented system.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisors Dr. Joey Paquet and Dr. Rachida Dssouli for their patient, guidance, support and valuable advices without which this thesis would not have been possible. Dr. Dssouli, you were a source of motivation and inspiration to me, thank you for your encouragement, unique kindness and for your time whenever I needed help despite your busy schedule. God bless you.

I am also grateful to Dr. May El Barachi and Dr. Nadjia Kara for their involvement, guidance, and help throughout this work. I also thank my fellow labmates especially Serguei A. Mokhov for their unique teamwork and the stimulating discussions we have had.

I would like to thank *castell*, the retired printer, who was always there for me in the sleepless nights I spent during my studies.

Last but not least, I would like to thank my greatest and dearest parents for their unlimited love and encouragement.

I would like to acknowledge the financial support of The Natural Sciences and Engineering Research Council of Canada and the Faculty of Engineering and Computer Science of Concordia University.

Contents

List of Figures

List of Tables

List of Algorithms and Listings

List of Abbreviations

API	Application Programing Interface
HTTP	Hypertext Transfer Protocol
InP	Infrastructure Provider
ISP	Internet Service Provider
JAXB	Java Architecture for XML Binding
NIC	Network Interface Controller
NVE	Network Virtualization Environment
PIP	Physical Infrastructure Provider
QoS	Quality of Service
REST	Representational State Transfer
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
SP	Service Provider
SSH	Secure Shell
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VIP	Virtual Infrastructure Provider
VMM	Virtual Machine Monitor
VM	Virtual Machine
VN	Virtual Network

VoIP Voice over Internet Protocol

VPN Virtual Private Network

WADL Web Application Description Language

WSDL Web Service Description Language

XAPI Xen Management API

XML eXtensible Markup Language

XSD XML Schema Definition

Chapter 1

Introduction

In this chapter we give an introduction to the research domain. We present an overview of the problem being addressed. Then, we discuss our contributions and the solution we propose.

1.1 Overview

The Internet is serving more than one billion people and has become a critical element of life that humans rely on to access information and do business. In just three decades, it has gained a tremendous success as worldwide communication medium. Started as a research project called the ARPANET in the late 1960's [7], the Internet, or "network of networks", was initially designed as a simple technology that enables two hosts to exchange packets. Then, it has evolved to support sending files and messages transfer. In the past recent years, advances in link technologies (such as fiber optics and wireless) as well as the wide variety of multimedia and sharing applications caused the Internet to explode in the number of users and the variety of applications and services offered. Thus, advances in mobility computing and the emergence of social networks increased the demand on the Internet usage. Despite its success, the increasing demand has demonstrated that the current Internet architecture created obstacles for introducing changes and innovations in the underlying networking

technologies. As the Internet continues to evolve, the innovation in network technologies and deployment of new services face many challenges such as scalability issues, new routing protocols, addressing, security and QoS. In contrast to the current Internet that was first designed for non-mobile hosts (having fixed locations), in just a very short period of time the mobile computing-related technologies went through an exponential progress.

Given this tremendous success, any technological changes in the Internet architecture could result in unintended consequences on the economy and overall Internet usage. Several ongoing research efforts are being conducted to redesign (i.e redefine) a new Internet architecture that is referred to Future Internet or next-generation networks [3, 5, 8], in response to the challenges and obstacles the current architecture is facing as well as promoting flexibility and diversifying services offered. As opposed to the current architecture, the Future Internet should allow co-existence of heterogeneous network designs and solutions (old and new), enable innovation, and address the challenges of the future related to the consumption services over the Internet. Consequently, many new architectures and paradigms such as Software Defined Networking (SDN) have emerged.

Network virtualization is considered as a promising way to overcome the limitations and fight the gradual ossification of the current Internet infrastructure. The network virtualization concept consists in the dynamic creation of several co-existing logical network instances (or virtual networks) over a shared physical network infrastructure [9].

The key concept of the Future Internet is to enable the coexistence of several network architectures in a multi-role-based environment. In such environment, several roles (i.e organizations) collaborate to offer distinct or similar services. Hence, the role of the traditional Internet Service Provider (ISP) should be decoupled and distribute its responsibilities and tasks among new entities: infrastructure provider (InP) who is responsible for managing the physical resources; and Service Provider (SP) who leases and aggregates resources (from one or more InP) on which he deploys end-to-end services (e.g., layer 3 VPNs, VoIP,conferencing services, etc) In order to leave the core network simple and dedicated to packet forwarding, the end-to-end principle claims that service's functionality

should not be deployed in the core Internet, but if possible, should be deployed on the end points [10].

To allow multiple networks to coexists, virtualization techniques have been considered as key enabler. Thus, the CABO project [3] was first to propose the use of virtualization to enable the creation of virtual networks over a shared physical network infrastructure. Essentially, virtualization is a mechanism that provides decoupling (separation) of the resulting services from the underlying physical environment. Network virtualization is intended to separate the data plane from control plane. This is expected to enhance the flexibility of the resulting networks, diversity and manageability of the offered services with quality of service capabilities.

1.2 Motivations

At the time of writing, the applications and services that are delivered over the Internet are rapidly evolving. Among such services are content distribution and multimedia service delivery (VoIP, video streaming). Moreover, several factors such as economies of scale, the need for lowering the IT costs, and the emergence of utility computing (pay only for what you need) have led to the adoption of a new software delivery model: Software as a Service (SaaS). In this model, software is consumed as utility and delivered over the Internet [11]. However, the current Internet architecture suffers from ossification and is ill-suited to support such types of application for many reasons [8, 12]. First, the underlying networking technologies did not evolve since the Internet inception. Second, the lack in support of the requirements needed to run those applications such as efficient routing, security, QoS as well as the support for mobile network-related features such as context-awareness, multi-homing, and seamless switching.

Internet ossification can be explained as the resistance of the current Internet architecture to support fundamental changes needed for deploying new network protocols, technologies and applications such as differentiated services, secure routing protocols, IP multicast. This creates significant barriers to innovation. Another issue to overcome the Internet impasse is

that by nature, the current Internet architecture relies on a single-role based business model: the Internet Service Provider (ISP) [1]. Consequently, the Internet infrastructure is shared among those ISPs where each ISP owns and manages its network infrastructure, and provides services to customers in a fierce competition environment based on self-interests and not quality of service. Competition and conflict of interest are issues among major stakeholders pose a barrier to the introduction of new and innovative network technologies. As a matter of fact, it is difficult to introduce changes to the current Internet architecture or adopt a new one without coordination and consensus between many stakeholders. At the time of writing, the deployment of necessary changes to support IPv6 that is happening at a slow pace consists in a good example to this problem. Radical changes to the Internet are needed as the demand on value-added services (e.g context-aware application) is rapidly growing. Remarkably, fierce competition between the major players in mobile computing has led to phenomenal advances in terms of innovation and services. Consequently, a large number of organizations that provide mobile-based services emerged. Enabling competition in networking technologies is perceived to have the same impact for building a better Internet [2].

To overcome the problem of Internet ossification, network virtualization has been considered a prominent solution which allows for enabling experimentation with new network architectures (or the Future Internet), and the deployment and test of new network services independently without disruptions [2, 8, 13–15]. Network virtualization is a promising and technically challenging concept, which enables the on-demand creation and provisioning of logical networks (or virtual networks) deployed over a shared physical network infrastructure. There are several motivations behind this concept, including cost-effective sharing of resources; customizable networking solutions; and the convergence of existing network infrastructures. The main idea behind network virtualization is to allow multiple heterogeneous and isolated network architectures (instances) to cohabit on a shared physical substrate. This is intended to provide better flexibility and manageability, as well as enhanced security allowing the deployment and test of customized services such as value-added services at low cost [3]. Network virtualization follows a new business model which

decouples the role of the traditional ISPs into independent roles: infrastructure provider (InP) who owns and manages the physical infrastructure (or substrate), and virtual network provider (VNP), who leases virtual resource from one or multiple InPs on which he creates virtual network(s) and offers end-to-end services. Offered as a service, a virtual network is provisioned on demand and the provisioning process consists in describing, discovering, matching, and allocating virtual resources [16]. In addition, a VNP selects the best virtual resources that are more suitable to satisfy its requirements. Precision and accuracy are critical to a successful selection of resources. Therefore, resources are evaluated and selected based on their static (functional) and dynamic (non-functional) attributes [17].

1.3 Thesis Objectives

Although network virtualization is seen as a promising solution for a more flexible use of the Internet, it is associated with many challenges related to the provisioning and operating virtual networks as well as the management of resource information (i.e., description and discovery).

For an infrastructure provider (InP) to enable other roles to discover and formulate their requirements in terms of resources, and for end users to discover the SPs' services of interest, there should be a standardized public interfaces for enabling role-to-role communication. These interfaces should also enable the programmatic management of the virtual resources [1].

Moreover, for SPs to request resources needed their desired virtual networks, there must be a mechanism that enables the dynamic discovery and selection of virtual resources that can be composed to form virtual networks. To achieve that task, there is a need for a formal and expressive information model which enables InPs to describe their resources and services, and facilitates information representation and sharing between the various roles/entities involved. In an attempt to address these challenges, we have set several goals.

First, to enable the effective interactions between various roles, suitable interfaces must be defined and standardized. Such interfaces would enable seamless interactions between the

existing various roles operating at different levels of the network virtualization environment hierarchy.

Second, a suitable framework is needed that enables the dynamic publication and discovery of available virtual resources. This framework would enable infrastructure providers to describe and advertise information about resources and the creation of virtual network(s) spanning across different administrative domains. Resource description should include static and dynamic attributes. However, the existing resource description languages focus on a specific aspect of network area and do not support the description of a virtual network as a whole.

The third goal is the investigation and elaboration of a distributed and scalable architecture that allows for publication and intelligent discovery of resources and services. This requires the identification of functional entities involved, the interfaces needed as well as the identification of interactions between the different entities.

1.4 Thesis Contributions

To solve the issues mentioned in the previous section 1.3, we propose a framework that enables the description as well as dynamic publication and discovery of resources in network virtualization environment(NVE) [9].
Our work focuses on modeling the resources and the publication and discovery of resource-related aspects as well as the interactions between various roles. We summarize the contribution of this thesis as follows:

- We propose a broker-based architecture for resource publication, discovery and negotiation in NVE that enables multiple roles to publish, discover and negotiate information about virtual resources. We then present the state of the art of resource discovery and selection in NVE.

- We build on the business model introduced in [6] and the broker-based architecture by proposing a multiservice, multi-role hierarchical information model, for network

virtualization environment. We then demonstrate the usage of this information model using a *secure content distribution scenario* that is realized using REST interfaces. Additionally, we present the relevant related work on network resource description and modeling in general and the ones proposed for NVE in particular.

- We implement a subset of the proposed architecture as a proof of concept prototype and to validate our proposed approach. We then extend the secure content scenario to illustrate and detail the interactions of the implemented components.

- We finally assess the performance of the implemented prototype and test the scalability of the overall system implementation.

1.5 Thesis Organization

The structure of this thesis is organized as follows. In the following chapter, we give a background information about virtualization, network virtualization environment and its business models as well as web services in general and RESTful web services in particular. In Chapter 3, we present the proposed framework for resource publication and discovery in NVE and a case study. We then present the related work on resource discovery. In Chapter 4, we present the proposed information model and we discuss the different information model that have been previously proposed for network resource description in general. In Chapter 5, we discuss the software architecture and setup of the implemented prototype. Then, we present the different interfaces used to interact with the implemented components. In Chapter 6, we evaluate the prototype by conducting performance and scalability tests and interpret the results. Finally, in the last chapter, we discuss the limitation of this work as well as suggestions for future work.

Chapter 2

Background and Related Work

2.1 Overview

Network virtualization is a quickly emerging paradigm, mainly, as an alternative to the current Internet architecture. It gained widespread popularity among, and has received the attention of, many researchers and companies due to the promising benefits and challenges it offers. New concepts and innovations are being adopted, among others, software defined networking, virtual network in data centers, etc.

In this chapter, first, we define the terms and concepts that are related to the work presented in this thesis. We then present the network virtualization concept, its environment and architecture as well as the related business models. We finally give an overview of review web services and resource oriented architecture.

2.2 Definitions of Terms

In this section we define the key terminology used throughout this thesis.

Resource – We use the term resource to refer to a manageable unit within a NVE context. A resource could be physical or virtual and it is associated with a set of characteristics such as processing power, storage, etc.

Physical resource – Or substrate node is a network element (network-capable electronic device), for instance, a network router, switch, server or computer.

Physical Infrastructure – Is a pool of physical resources that are intended to be virtualized and used on demand.

Virtual Resource – Virtual resource or virtual device is a logical entity that is created using some virtualization techniques and hosted on a physical resource (e.g a virtual machine or a virtual router). Thus, many virtual resources can co-exist on, and share the computational resources of, the same physical resource . Virtual resources can be aggregated from two or more providers forming what is called a federation.

Service – We use the term service to refer to network services or application offered by different providers. Such as firewall application, load balancers, VoIP, conferencing services and so on.

Virtual Network – Consists of at least two virtual nodes connected together by a virtual link. A virtual network is deployed on top of, and inherits the same properties as, a physical one. Thus, a virtual network can span across multiple domains.

Resource Provisioning – The term provisioning refers to the set of processes that are performed to allocate resources or services to consumers upon request. The provisioning process often includes steps related to resource description, discovery and negotiation. Negotiation enables different entities to reach an agreement on the quality and the guarantee of the service in question.

2.3 Virtualization

In essence, virtualization is the process by which an additional logical layer (software) is added on top of on existing system so that the resulting system reflects the underlying one while exposing its services and mimicking its behavior. The services and functionalities of the underlying system are exposed so the user get the illusion as if it was using the real system. Virtualization is not a new concept and was first introduced in the early 1960s by IBM for the third generation computers (i.e mainframes) as a mechanism to divide the

computational resources for different applications [18]. However, the use of virtualization has declined around the early 80s as there was a big shift in terms of processing power and due to the lack stability of computers' architecture in general.

In 2000s, several factors and motivations such as advances in computer hardware, under-used machines, the need for server consolidation, cost saving, etc, have led to the "renaissance" of virtualization and mainly its mainstream adoption for hardware virtualization. Moreover, it has been applied to multiple computing areas such as storage, applications, network, and hardware. Although virtualization offers many benefits such as flexibility, effective use of resources, programmatic allocation of resources, isolation and security, on the other hand, there are several drawbacks associated with it such as performance overhead and complexity, as well as the lack of supporting all type of applications. Among today's major virtualization technologies products are Microsoft Hyper-V [19], VMWare vSphere [20], Citrix (XenServer) [21].

2.3.1 Types of Virtualization

Virtualization can be applied to hardware (hardware virtualization) as well as to software (applications virtualization). In general, we distinguish several virtualization categories, such as hardware-level (sever and router virtualization), operating system-level (e.g Jails), high-level programming language virtual machines (e.g Java VM, .NET CLR), etc.

However, in this thesis, we are concerned about the following most common virtualization techniques for hardware server virtualization:

- **Full virtualization:** in this technique, the guest operating system is used as is without modification. However, this results in poor performance due to the lack of direct access to the underlying services and poor I/O performance. This allows older and legacy systems to run on newer and more efficient machines [22].

- **Para-virtulization:** this technique requires a modified guest OS that achieves near-native system performance (with total around 5% overhead) [23]. The

modifications are made to improve performance by making the guest "aware" that it is being virtualized [24]. As opposed to full-virtualization (e.g VMWare, KVM), para-virtualization-based solutions offer scalability and offer notable performance improvement that is near to native one with 0.5 to 8% performance overhead [23].

Both approaches requires what so-called a virtual machine monitor (Hypervisor). The hypervisor is a type of OS that allows multiple operating systems, called guest OS, to run simultaneously on a single physical machine.

2.4 Cloud Computing

At the time of writing, there is no consensus nor a standard definition to define Cloud Computing (CC). It can be seen as a new paradigm where computational resources (e.g servers, storage, networks) can be dynamically requested, customized and provisioned on demand which are provisioned and managed from a single point of management by a service provider. For a more technical definition, it is a distributed system consisting of a large pool of virtualized resources consumed on demand on a pay-per-use basis. Several factors such as advances in computer hardware, under-used machines, economic crisis, reduction of operational cost and increased demand on computational resources have led to the adoption of this new paradigm. Moreover, today's computing environments are changing and shifting towards a service-based model where everything, from a server to a software, is offered as a service. More and more new paradigms are being introduced that are based on concepts such as on utility-based usage, multi-tenancy and cost-effective sharing of resources [25]. Consequently, CC inherits many pre-existing technical concepts (such as distributed computing, grid computing, and virtualization) and business-related concepts (such as utility-based pricing, multi-tenancy, cost-effective and efficient use of hardware resources) [11].

2.4.1 Cloud Computing Business Model

The business model related to CC environment is centered on the following for roles:

Infrastructure providers – Provide their infrastructure (i.e hardware and computational resources) as a service (IaaS) to service providers. Among the major infrastructure providers are Amazon [26], Google [27], IBM [28] and recently Microsoft [29].

Platform providers – Provide development platform as a service (PaaS) to develop customized cloud-based services. Example of these platform are IBM SmartCloud, Amazon Elastic Beanstalk and Google App Engine.

Service Providers – Are organizations offering services to consumers. A service can be a software (SaaS) or anything (*aaS) and offers value to the consumer and can be composite: composed of two or more services aggregated together from one or more service provider. A service provider relies on the services provided by infrastructure providers in terms of hardware and computational resources.

Consumer –Or end user is the buyer of a service that is provided directly by a service provider or a platform provider.

2.5 Network Virtualization

Virtualization has been around for a long time and it consists of the introduction of an abstraction layer on top of physical resources which gives the impression to the user as if he is using those physical ones. The concept of network virtualization, which enables multiple logical networks to co-exist, is not new and has evolved since its first introduction [1, 8].

Past virtual networking techniques/technologies are Virtual Local Area Networks VLANs, Virtual Private Networks (VPN), active programmable area networks and overlay networks and the term "virtual network" is used to refer to those technologies. However, the term network virtualization that is used throughout this thesis refers to a new network virtualization concept where network virtualization is not limited in scope (e.g, layer 2 for L2 VPN) as opposed to the aforementioned technologies [30].

2.5.1 Overlay Networks

Overlay networks are logical (implemented in the application layer) networks built on top of one or more physical networks . In its early days, the Internet started as overlay that was deployed on top of the telecommunication network. The key advantage of overlay networks is that they do not require any changes to be done on the physical networks, nor do they affect their functioning [31]. This has led to use overlays extensively as successful solution to test and deploy new features and fixes in the Internet. Moreover, many overlay-based testbed projects have been initiated such PlanetLab [32], X-Bone [33] and VINI [34] that are used as testbeds to design and test new architectures, study computer viruses and propose applications that address many issues (such as Internet routing, protection from denial of service attacks, etc). Although overlays introduce flexibility and offer the possibility to build logical networks independent from physical ones, most overlays are designed and built at the application layer (on top of IP). Therefore, they cannot solve the limitations of the current Internet architecture.

2.5.2 The Network Virtualization Environment

As shown in Figure 1 The network virtualization environment(NVE) consists of one or more heterogeneous virtual networks co-existing on one or more virtualized platforms. A virtual network (VN) consists of the basic entity in such environment. VNs within NVE use different network architectures and protocols and provided as a service by various roles (called virtual network providers). The network virtualization concept enables multiple, isolated, virtual networks running simultaneously to share, and co-exist on, the same physical resources. To achieve this concept, the physical resources are divided into slices (partitioned) using some virtualization technologies. The resulting slices are called virtual resources consist of the key building blocks of a virtual network. Virtual resources are requested on demand

and aggregated to create a VN that is offered as a service. Many network virtualization research projects have been initiated. Such projects are: CABO (Concurrent Architectures are Better than One) [3], 4WARD [4], Nouveau [35,36]. The aforementioned projects follow a "pluralist" philosophy which states that networks should be fully virtualized and network services should be separated from their underlying infrastructure [13].

The idea behind network virtualization is not new. Several approaches have been introduced to put in place what is called virtual networks (VPN, VLAN). Network virtualization requires virtualized platforms and relies on two main components: node virtualization and link virtualization [1,8]. However, network virtualization goes one step further by taking advantage of the benefits of platform virtualization such as the cost-effective use of resources, network isolation, optimization and programmability as well as the dynamic provisioning of resources which enhance scalability.

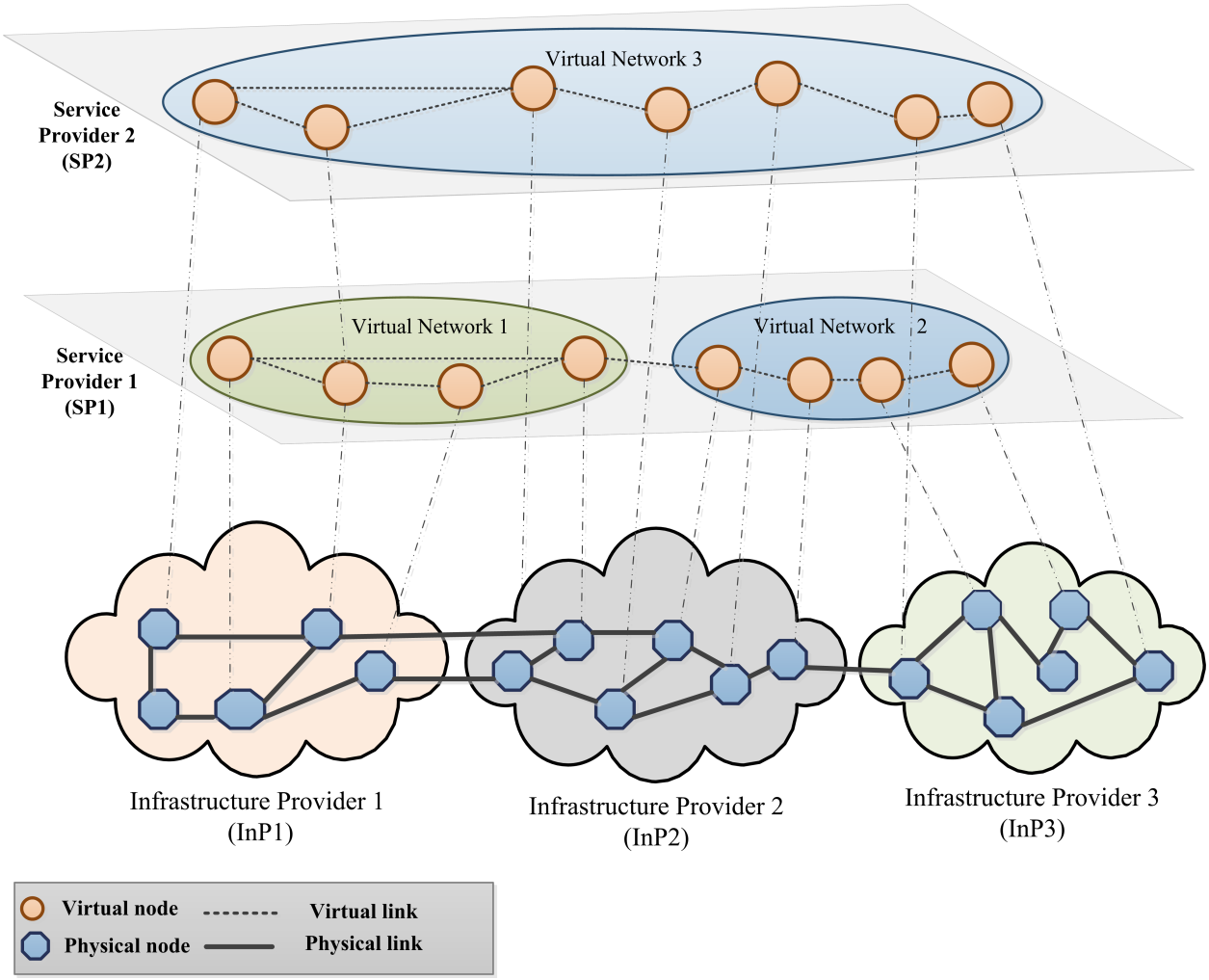

Figure 1: The Network Virtualization Environmentadapted from [1]

2.5.2.1 Business Models

From a business perspective, one of the goals of network virtualization is to introduce flexibility and innovation by decoupling the role of the traditional Internet service provider (ISP) into multiple independent entities. Figure 2 depicts the initial business model for NVE as proposed in [2, 3]. This model introduces two main roles:

Infrastructure Provider – Who owns and manages the physical infrastructure. He is also responsible for creating virtual resources by partitioning his physical resources into isolated slices using some virtualization technology.

Service Provider – Leases and aggregates virtual resources from multiple InPs to create

virtual networks. Furthermore, the service provider is also responsible for offering end-to-end services to end users .

End Users – in NVE are the consumer of the services offered by SPs. They are similar to end users in the traditional Internet model; however, they have a large number of services to choose from. This is due to the multiple virtual networks offered by competing service providers [1].

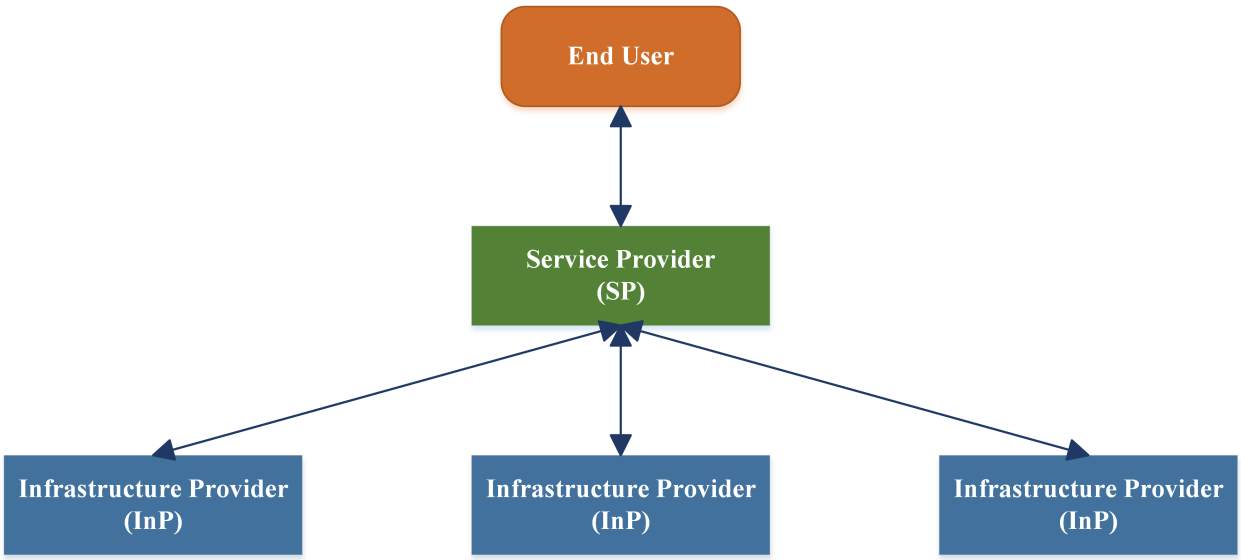

Figure 2: Main roles of NVE [2, 3]

Another model has been proposed as part of the 4WARD model, which consists of one of the main research projects on network virtualization [4, 5]. The 4WARD model extends the model shown in Figure 3, and defines two more roles:

Virtual Network Provider (VNP) – finds and aggregates the optimal virtual resources from one or more InPs. These aggregated virtual resources are leased from multiple InPs to fulfill the virtual network operator's request. The VNP does not provide a network but only the virtual resources on which the virtual network operator deploys the protocols to create a virtual network.

Virtual Network Operator (VNO) – creates the virtual network over the virtual resources that were previously aggregated by the VNP. He deploys the required protocols

stack and network architecture over the newly created virtual network. In addition, the VNO is responsible for controlling, managing and maintaining the virtual network. In this model, the SP has no direct interaction with the infrastructure provider. However, he deals with the VNO as if it was the InP to deploy the end-to-end services he offers.

As matter of fact, the success of NVE model depends on the collaboration and interaction between various business entities. Each business entity not necessarily belongs to a particular category of roles, provides a distinct service that is intended to be used by another entity within the NVE. Moreover, several business entities provide services, when combined together; enable the creation of VN on which an end-users service is deployed. In this thesis, we use the term role to refer to a business organization doing business in NVE.

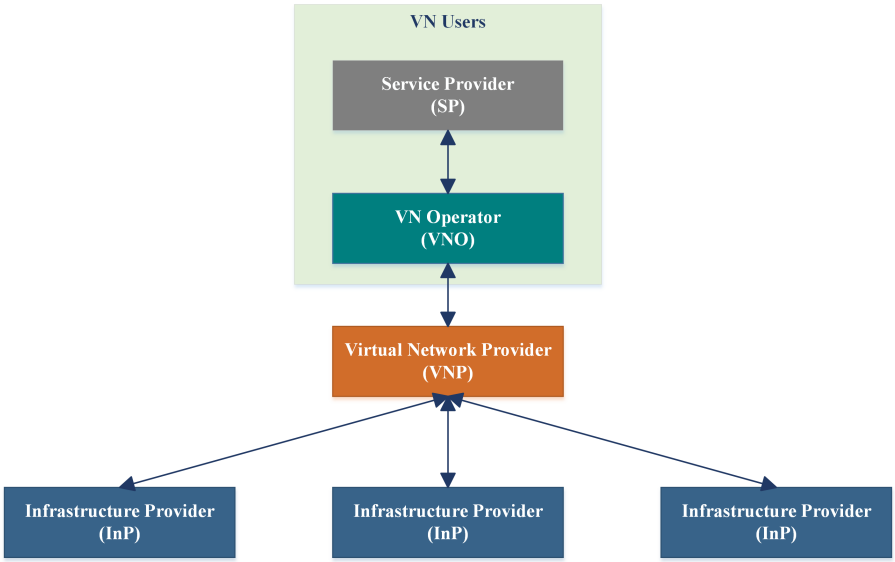

Figure 3: The business model proposed for the 4WARD project [4, 5]

2.5.2.2 Service-Oriented Business Model

Inspired by the TINA [37] and the web services composition models, El Barachi et al. have introduced a new service-oriented hierarchical business model for network virtualization environment in [6,38]. The proposed business model is shown in [6,38]. It puts the emphasis on the notion of services, hence, four classes of services are defined, namely: *Essential Services* constituting mandatory services needed for the basic operation of the network (i.e.

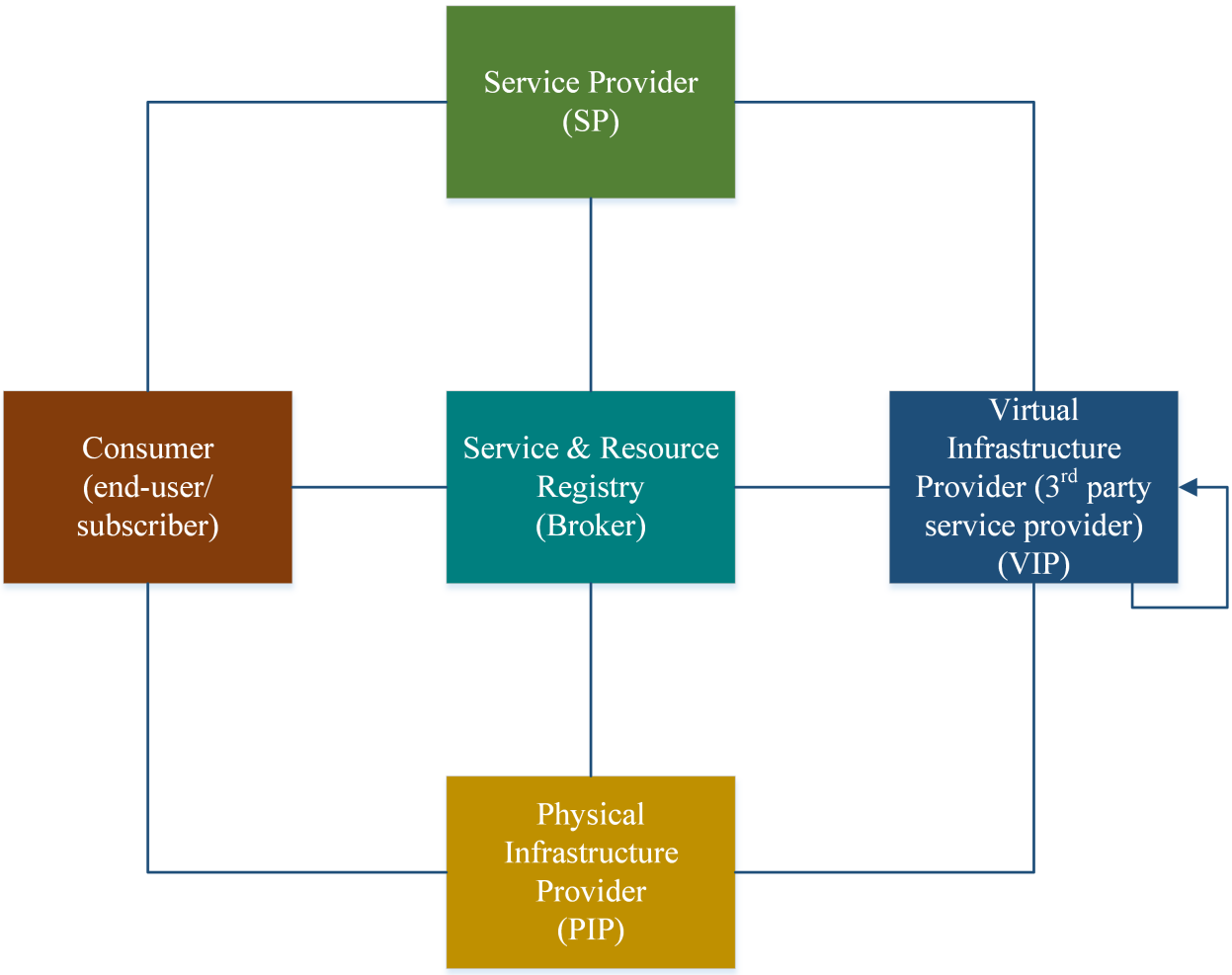

Figure 4: The service-oriented business model as proposed in [6]

routing/transport services); *Service Enablers* consisting of the common functions needed to support the operation of end-user services (e.g. session/subscription management, charging, security, and QoS management); *Service Building Blocks* acting as elementary services that can be used/combined to form more complex services (e.g. presence and call control); and *End User Services* constituting the value-added services offered to users.

In this model, five distinct roles were introduced which we briefly detail as follows: ***Physical Infrastructure provider (PIP)*** – Owns and manages a physical network infrastructure (or a substrate network); splits the resources into isolated slices; and describes and advertises the virtual resources being offered.

Virtual Infrastructure Provider (VIP) – Aggregates resources from one or more PIPs, instantiates and operates a virtual network on which he deploys service enablers.

Service Provider (SP) – offers value-added services to subscribers with whom he has a business agreement. The SP aggregates multiple service building blocks to form a composite service.

Broker – Or Services and Resources Registry (SRR): is a repository where information about the advertised virtual resources is stored. It puts all the parties in contact by providing relevant information to find other parties and the services/resources they offer.

Consumer – acts as end user and service subscriber and uses the value added services provided by SPs.

Figure 4, illustrates the relationships that a role could have in NVE. Roles are business organizations that interoperate and collaborate with each other in order to consume/offer resources and services and exchange information related to these resource/services. Therefore, a role can be a resource provider and service provider at the same time. Being the former, a role offers, manages and accesses virtualized resources. Whereas a service provider offers, manages, and sometimes subscribe to network services. In fact, network services are mapped to networked resources. Roles are distributed and loosely coupled entities. In consequence, to enable interactions between them, programmable interfaces are needed. Thus, just like web services, network services can be published, dynamically discovered, used

and composed.

2.5.3 Virtual Network Embedding Process

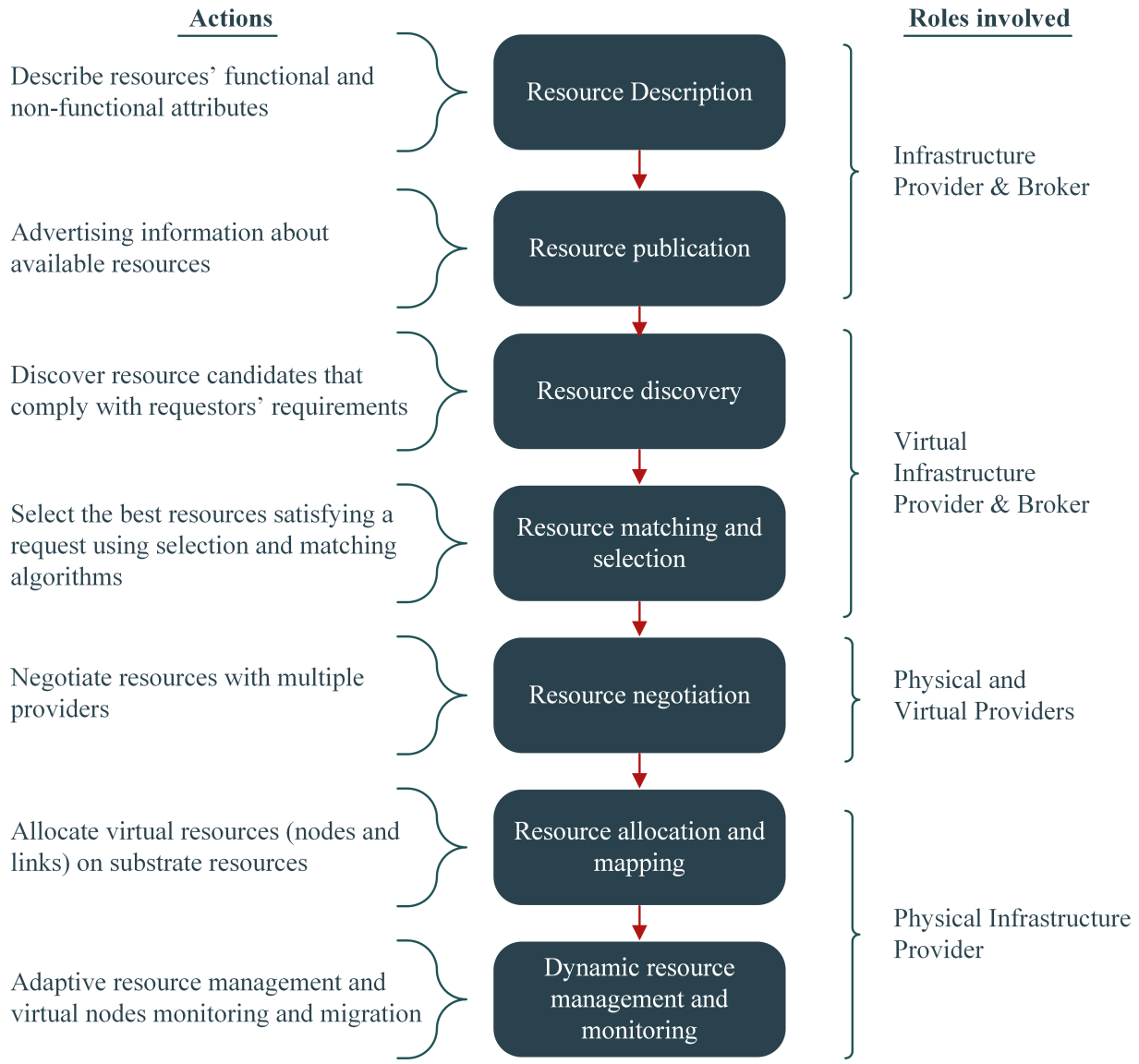

Figure 5: The virtual network embedding process

As shown in Figure 5, the virtual network embedding process involves several steps and requires the collaboration between various roles. We briefly detail the steps as follows:

Resource Description – In NVE, multiple infrastructure providers offer resources to be

leased on demand. Describing the offered resources is crucial in NVE, however, it is not new to the network field. Resource description (or information modeling) involves describing the fine-grained functionality of network resources (i.e their characteristics and capabilities) [16]. This facilitates information retrieval related so that the resources can be discovered and selected precisely and efficiently by other roles wishing to lease the resources offered by the infrastructure providers. In other words, resource description can be defined as the specification of the functional and non-functional attributes of resources using a description language.

Resource Publication – The process of resource publication in NVE is similar to the process of publishing a web service's description. It involves the steps performed by an infrastructure provider to advertise (i.e share information) the virtual resource description he offers. According to the literature, the description of the resources are registered in public discovery framework [17], or repositories, so that they can be discovered by VN requestors.

Resource Discovery – Resource discovery requires that information about the virtual resources offered by multiple providers must be available. This process consists of seeking relevant resources/services of interest according to their characteristics and selecting the best infrastructure providers [17, 39]. A VN request is used to formulate the characteristics of the requirements wanted resources should have. Such requirements are later taken into consideration by the resource selection and matching process.

Resource Selection – This process consists in finding from a set of potential resources the best resources whose characteristics correspond to the requirements specified in the discovery request. This involves matching the attributes of the candidate resources with the one specified in the discovery request. Some selection algorithms can be used such as hierarchical resource clustering using dendrogram [16, 17].

Resource Negotiation – Resource negotiation is considered as important process in NVE that enables service provider to negotiate with multiple providers in order to select the best one. This involves the negotiation of the capabilities of the requested resources and the related quality of service scheme [40].

Resource Allocation Mapping – Resource allocation or resource provisioning [8], also referred to as VN embedding in the literature, is the process of reserving physical resources to virtual resources (such as nodes and links). Resource allocation is performed by an infrastructure provider.

Dynamic Resource Management – This involves adaptive management and monitoring of the allocated resources after their creation [41, 42]. As part of this process, virtual nodes can be migrated, hence, the virtual topology can be dynamically adapted.

2.5.4 Virtual Network Applications and Services

Much like server virtualization that enables multiple virtual machines to run on the same physical server, network virtualization enables multiple virtual network sharing the same physical resources. Hence, different protocol and communication technology can be used to offer different kind of sophisticated applications which require high. Example of such applications are conferencing services, content delivery networks (video streaming) [43], Internet Protocol Television (IPTV) [44], etc.

2.6 Web Services

Web services are software components exposed to the web via public interfaces enabling them to be invoked remotely. For them to be discovered and consumed, web service description languages such as WSDL and WADL are used to describe services' functionality and capabilities [45, 46]. Such description includes the operations a service offers, the data being transmitted and communication protocol supported, and so forth. Web services were first introduced around 1998 to cope with interoperability problem encountered with early-distributed system technology such as the Common Object Request Broker (CORBA). The Service-oriented architecture (SOA) was the first architecture proposed for building web services and automating business processes and enhancing systems' productivity and interoperability between them. SOAP-based web services (also known as big web

services), which are built upon the SOA architecture, have been widely adopted by several organizations for integrating enterprise applications as well as implementing and composing business processes [47].

2.6.1 RESTful Web services

Representational State Transfer (REST) is not an architecture but rather an architectural style. It was introduced by one of the principal HTTP authors, Roy Fielding in his dissertation [48,49] in 2000. The main idea of REST revolves around the notion of resources that need to be used and addressed. A resource can be anything exposed to the web such as any application's component, file on disk, a database recode and so on. REST web services rely on HTTP protocol as communication enabler and take advantage of its reliability to build a large-scale distributed hypermedia systems. As opposed to SOAP-based web services, REST web services are lightweight services, which make them easy to discover and consume by any application that is able to send HTTP requests from any kind of client (mobile, browser, wireless sensor network). Consequently, in recent years, REST has attracted the attention of large number of companies, mainly the major Internet player, to expose their data and resources. In essence, REST architecture style is based on the following principles:

Resource Identification. Similarly to a web page, a REST web service is a collection of resources. Each resource is uniquely identified by a URI (Uniform Resource Identification) which enables global resources accessing and service discovery.

Uniform interface. Resources are manipulated and accessed on CRUD (create, read, update and deleted) basis via HTTP's standardized operations: PUT, GET, POST and DELETE.

Self-descriptive messages. REST decouples resources from their representation. Hence, a single resource can be represented in format (XML, JSON, Text). Each message contains the metadata that describes the meaning of the message and the information needed on how to process the message (e.g mime type, HTTP operation used, etc).

Hypermedia as the engine of application state (HATEOAS). This principle enables

to build hypermedia-driven applications: the client of REST web services can follow the links contained in resource's representation to go to next state (change resource's current state). Thus, REST inherits some of HTTP's properties, hence, interaction with a resource is a stateless. However, stateful interactions are possible through hyperlinks.

REST web series are often referred to REST API. Much like a web site, an API has a base URI (root), for example, http://example.com/resources/. Moreover, the API specifies the format of the data exchanged is determined using Internet media type (previously called mime type) and the set of operations supported using HTTP methods (e.g GET, PUT, POST or DELETE). In this work, we use REST web services in implementing public interfaces that enable interaction and interoperability between various roles. A more detailed discussion about the use of such services is presented in Chapter 5.

2.7 Summary

The network virtualization is not a new concept. However, the introduction of virtualization technologies into the networks enabled has enormously contributed to the undergoing research effort to define a new diversified Internet architecture (i.e., Future Internet). Virtualization is seen as a promising solution that enables to divide physical resources into slices (i.e., virtual resources) which results in enabling cost-effective usage and sharing of resources as well as enabling flexibility and isolation among the resulted virtual machines. In this chapter, we first defined the terminology and terms used throughout this thesis. We first discussed virtual networks in general and presented the concept of network virtualization. In particular, we presented the notion of network virtualization environment (NVE) and the related business models as well as the virtual network embedding process. Moreover, we gave a brief introduction to web services technologies that we envisage to use as public interfaces enabling interactions between roles. The following chapter presents the framework for dynamic resource publication and discovery we propose.

Chapter 3

A Framework for Resource Publication and Discovery in Network Virtualization Environment

3.1 Introduction

We have discussed in the previous chapter the NVE and the related concepts and business models. In this chapter, we present the framework we propose for resource publication and discovery in NVE. We first survey the existing work on resource publication and discovery in NVE. We discuss publication and discovery mechanisms of information about network-related resources and the requirements such framework should have. Then, we present a high-level overview of the framework architecture and describe a case study as well as detail a scenario showing how this solution would be used.

3.2 Related Work to Resource Discovery and Selection in Network Virtualization Environment

In a federated network virtualization environment (similar resource being offered by many providers), resource discovery process consists of finding the available resources that are capable of providing a service according to a set of requirements describing the characteristics (i.e capabilities) of each wanted resource. It is a critical process for efficient allocation and selection of reliable resources. In addition, due to the heterogeneous nature and the dynamic ability of the resources, when selecting resource candidates, their functional and non-functional characteristics have to be taken into consideration. Several approaches have been proposed for discovering resources in distributed computing environments (e.g computational grids, cloud computing, peer-to-peer networks). Such approaches employ many well-known techniques [50], namely: distributed indexing (used in peer-to-peer networks), UDDI registries that are used in soap-based web services, broker-based repositories, resource classification and clustering [51] and agent-based.

Despite its importance in the VN embedding process, little work has been done on resource discovery in NVE. In [16], a resource discovery framework has been proposed for the 4WARD [4] model in which three parties are involved: PIP, VN provider, and customer. Resources are described using XML documents. This framework uses clustering techniques to manage arrange and resource information into dendograms (a tree-like data structure) which facilitates the resource matching and selection process. While only resource's functional attributes are advertised and stored in external repositories, non-functional attributes are constantly updated and kept in local repositories (are not made public) because of the possible overhead caused by the continuous monitoring of network resources. Moreover, this work considers only the case where a single VNP having to select the best PIP in order to embed each VN request. Another point is that in NVE multiple PIPs can offer the same resources (having similarities in their attributes), which implies that a VNP would deal with a single PIP at once.

Authors in [52] extended [16] to address the problem of virtual network embedding across multiple PIPs and to enhance the resource discovery process by reducing the search range and cost. They proposed a hierarchical virtual resource organization framework supporting multiple InPs that rely on using Local Management nodes to store static (functional) attributes and arranging them, at first, into conceptual clusters called Micro Clusters (MiCs) at the PIP level. Furthermore, a Cluster Index Server is used to aggregate and organize the MiCs belonging to various PIPs having the same root attribute resulting in a Macro Cluster (MaC). In this approach, dynamic (time variant) attributes such as the residual capacity of a substrate link is stored in the local management nodes. To benefit from WSDL's support for the dynamical update of services, authors in [53] proposed a WSDL-based resource provisioning framework for NVE. In this work, resource descriptions are integrated into WSDL documents. Thus, with the help of local agents deployed on local substrate networks, WSDL documents containing resource description are dynamically generated and published to UDDI registries. For a VNP (Virtual Network Provider) to select the candidate resources, a search in all the UDDI registries is required. In this framework, the selection process relies on parsing the information contained in WSDL, and uses the greedy and shortest path algorithms to retrieve the necessary information. Aiming at enhancing the efficiency of the resource selection process by considering the dynamic attributes, the Aggregation-based Discovery for Virtual Network Environments (ADVNE) is proposed in [39]. However, to minimize the continuous monitoring overhead of each single attribute (by reducing the number of monitoring messages circulation over the network), the authors propose an approach which consists of calculating the aggregate of the monitored attributes instead. In this approach, each PIP disposes a monitoring agent that monitors and calculates the aggregation values which will be later retrieved on-demand by the discovery module during the selection process.

To the best of our knowledge, the work that has addressed resource discovery in NVE focuses only on functional attributes. However, only authors in [39] take into consideration

non-functional attributes in the resource discovery process. We argue that considering only functional attributes in the selection process may lead to inaccurate selection. Furthermore, when similar resources are offered by many PIPs, the discovery process become crucial and the most appropriate resources have to be selected based on both functional and non-functional attributes. Although advertising non-functional attributes may generate some overhead due to the extra required processing, many solutions can be used to cope with this problem such as efficient monitoring techniques.

3.3 Business Scenario

A service provider (SP) wishes to offer a secure and QoS-enabled content distribution service for customers. Such a service is intended to be consumed on-demand; hence, the resources on which the service will be deployed should not be limited (i.e elastic) and dedicated only to the service in question. The SP would need to deploy its service over a scalable virtual network so that, if needed, additional resources can be easily requested and added to enhance the overall network's performance. In addition to having its own isolated virtual network, the instantiated virtual network would benefit from having additional value added properties such as enhanced security, and efficient routing-aware services as well as different QoS scheme support. At first, the SP would contact a VIP offering virtual resources of interest. He would send a VN request with details related to the required resources. The VIP processes the VN request, allocates the required resources in collaboration with the PIP, and finally grant access the SP access to the newly instantiated virtual network.

A more detailed scenario showing the interaction and the message exchanges between the involved roles are shown in Section 3.6, but first we introduce the architecture of the proposed framework in the next section.

3.4 Requirements for Dynamic Resource Publication and Discovery in Network Virtualization Environment

The designed architecture should be consistent and fit with the business model presented in Section 2.5.2.2, therefore such architecture should meet the following requirements:

- **Brokerage role.** To foster interoperability among various roles, the framework should enable seamless interaction between different virtual networking related roles/parties (i.e., Physical Infrastructure Provider, Virtual Infrastructure Provider, Service Provider, and Consumer) for the dynamic discovery of other parties and the services and resources they offer. The designed architecture should be consistent and fit with our proposed business model.

- **Suitable for network virtualization environment.** Distributed, self-managed/self-organized, and scalable in terms of the amount of information handled, in order to be suitable for the dynamic and elastic nature of virtual networks.

- **Information Management.** Another key requirement is that the framework should enable efficient management of information related to various physical/ virtual resources and services that may be offered in network virtualization environment. Information about resources should be well organized so that it can be quickly retrieved on demand.

  This requirement is multifold, therefore, is associated with the following sub-requirements that are also related to information management and organization of resources, namely:

  - **Information acquisition.** And update as well as the dynamic tracking and monitoring of resources and services' status information.

- **Information modeling.** Relies on a formal and expressive information model to facilitate information representation and sharing. This model should enable the description of the functional and non-functional aspects of available resources and services, as well as domain related semantics (cross-domains). Instances of this model (data models) should be used in the interaction between different roles/components within the framework.
- **Information dissemination.** Relies on standard protocols and suitable interaction models (e.g. publish/subscribe, polling, tuple spaces) for efficient information exchange across domains. Synchronous and asynchronous modes of communication should be supported to achieve flexibility.
- **Resource clustering and ranking.** Supports resource/service ranking and categorization, and a modular/layered design, to ensure efficiency of operations.
- **Support for service composition.** The framework should enable service composition and facilitate the composition of resources/services.

3.5 Broker-based Framework for Resource Publication and Discovery

Figure 6 illustrates the system architecture of the proposed framework. We selected a broker-based approach to cope with the complexity of publication and discovery of resources and services in network virtualization environment. The main objective of our work is to find an efficient solution that enables seamless interactions and collaborations between various roles. Within this framework, Physical Infrastructure Providers (PIP) are substrate resource suppliers who advertise (i.e., publish) the description of the resources offered into the Broker's service and resource repository (SRR). Virtual Infrastructure Providers (VIP) are virtual resource providers that discover the resources needed to instantiate a virtual network, and negotiate these resources with the selected potential PIPs. Moreover, VIPs request PIPs to instantiate VNs on which they deploy network services. In turn, Service

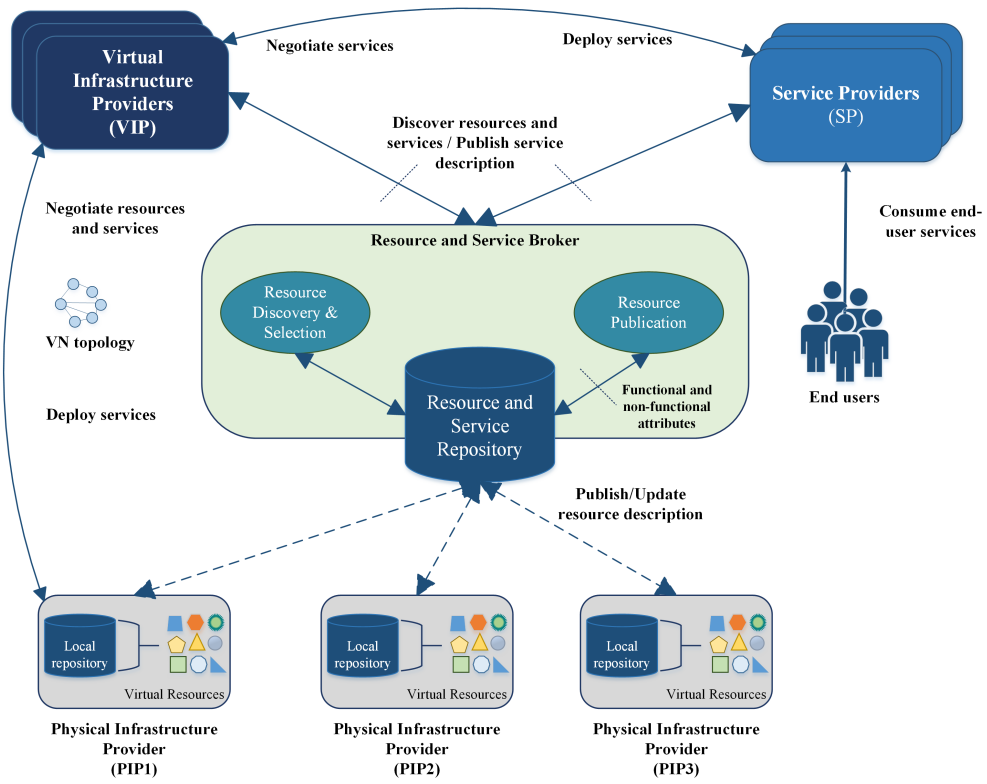

Figure 6: System architecture of the proposed framework for resource publication and discovery

Providers (SP) are end-to-end service providers who require and discover virtual networks on which they deploy the services they offer. Additionally, they negotiate the selected services with the appropriate VIPs. A more detailed description of the functions of this framework is presented in the following section.

3.5.1 Overall Architecture

3.5.1.1 Introduction

Figure 7 gives a high-level overview of the proposed framework architecture that is broker-based, multi-level (layered) and composed of a set of loosely coupled components. The PIP is represented at the Physical Level, in turn, the VIP is represented at the First Virtual Level and finally the SP represented at the second Virtual Level. Consequently, roles depend on

each other in performing the virtual network provisioning process. We selected a resource-broker approach to cope with the complexity of managing and organizing information about resources [54]. We introduce a resource and service broker which serves as mediator while coordinating the communication between various roles.

The resource broker allows not only roles to publish their resources and discover other roles' resources, but also provides the ability to select the most appropriate resource based on particular characteristics and constraints. Thus, it manages the inventory of federated resources in the resource and service registry (SRR), which holds well-defined static and dynamic resource properties. Both functional and non-functional attributes are advertised and stored in the (SRR). Additionally, many providers (VIP, SP) can discover other roles and the resources/services they offer through the broker's services. Upon receiving a resource discovery request, the broker selects the most appropriate resources that comply with the requirements as formulated in the request, and returns to the requestor the list of the candidate resources. Roles, in turn, can perform another selection stage in order to refine the list based on some local preferences (such as QoS, cost). Prior resource allocation, and to reach an agreement on the selected resources, roles negotiate resources' capabilities (such as price, availabilities, QoS-related parameters). At each level, we find local information sources (repositories) that the respective role uses to manage information about resources. However, only the information about resources that are intended to be offered are published into the resource and service broker. In this architecture, communication between layers is bidirectional and can be performed through a standardized connectivity provider (e.g public interfaces web services).

3.5.1.2 Components Description

The broker is composed of two main components. We distinguish two key services involved in the resource publication and discovery process, namely:
(a) *Publication Service* enables the publication (registration), updating and deletion of information about resources; (b) *Discovery Selection and Ranking Service* receives as input

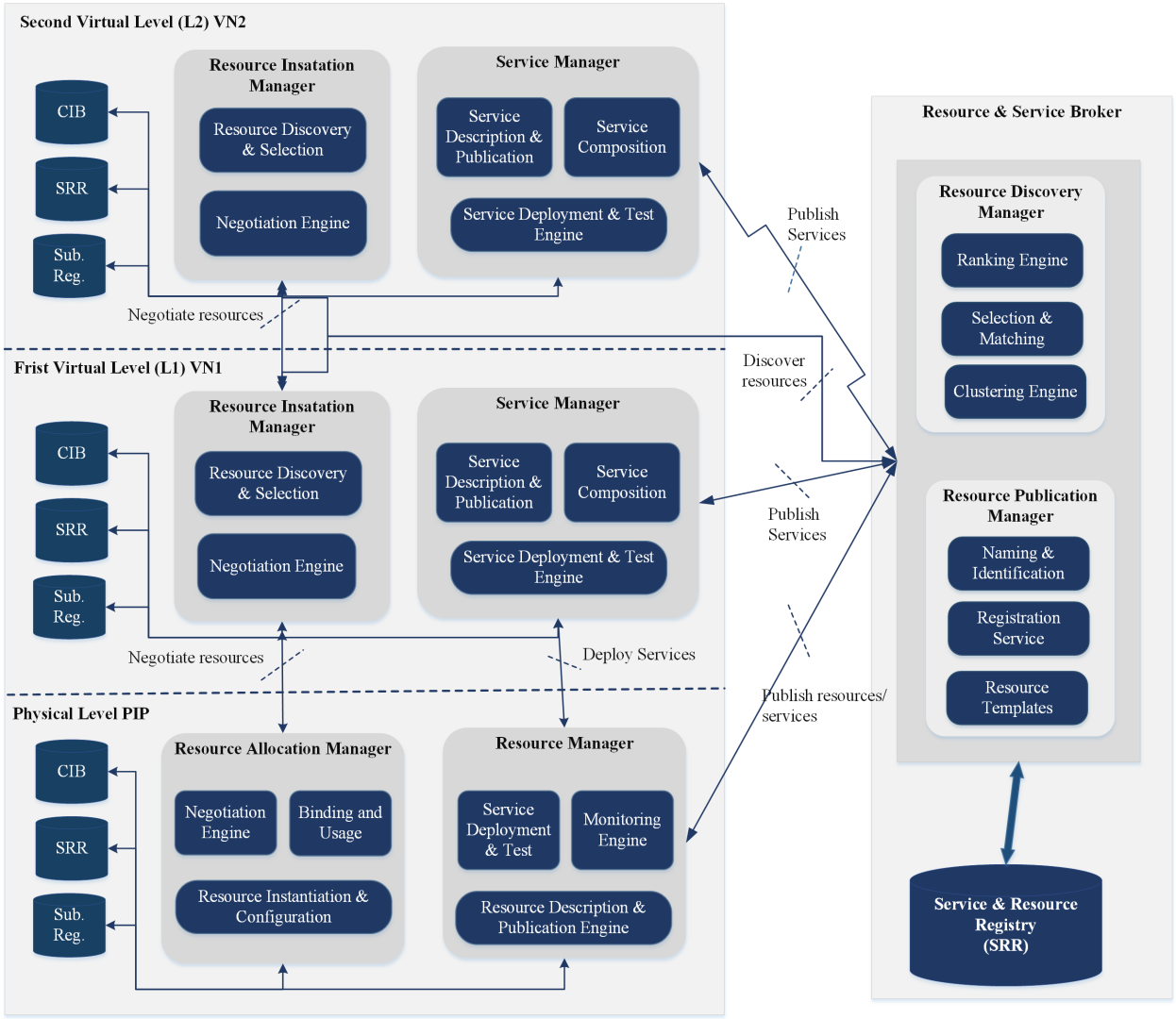

Figure 7: High-level overview of the proposed architecture

a request containing the description/requirements of resources of interest along with QoS constraints. While taking into consideration the resources' rank and the specified constraints (i.e QoS, cost, etc.), it selects the most appropriate resources that satisfy the request, and returns, as a result, the list of matching resources.

The *Ranking Engine* evaluates the popularity among similar resources and attributes a rank to each resource each time it is selected. This rank could be based on their usage, functional and non-functional characteristics (such as availability, uptime, cost and QoS, etc.). Furthermore, to facilitate resource selection, the *Clustering Engine* arranges information about resources contained in the SRR into clusters (grouping resources having similarities). (c) Following a well-defined naming scheme, the *Identification and Naming* service is responsible for dynamically instantiating a name (unique identifier) for each resource registered in the SRR. Because in a federated virtual resources environment many providers could offer the same resource; a unique identifier is needed to distinguish one resource from another. (d) *Templates Service* provides the different roles with an up-to-date template for describing resources or network services. The first layer of the hierarchy (L1) provides components for describing, publishing and instantiating resources as well as negotiating resources with other roles. The second and the top-level layers components that are responsible for describing, deploying and publishing network services. L1 contains components grouped into the following sub-systems: The *Resource Manager* (RM) As a whole, it handles the management and publication of resources and encompasses five components: The *Description and Publication Engine* consisting of a key enabler of the resource publication process, it enables a PIP to describe the resources he offers using an instance (i.e. a document) of the information model; and validates the generated instances to ensure data consistency and their conformance with the information model. Furthermore, it interacts with the broker (detailed below) in order to publish, update or delete information about resources.

The *Service deployment and Test Engine* enables the PIP to deploy and test essential services such as routing or transport services. Monitoring Engine monitors the status of the

allocated resources, and the links connecting the virtual nodes. Additionally, it continuously collects information about resources' dynamic properties for generating statistics purposes. The *Resource Allocation Manager* (RAM) coordinates all the steps involved in the resource allocation process (i.e. negotiation, instantiation, allocation and binding) and consists of the following components: The *Resource Negotiation* handles and coordinates the resource negotiation process with a given virtual layer. The *Resource Instantiation and Configuration* is responsible for the "slicing" of physical resources. It handles the instantiation request, and enables the creation and configuration of virtual resources as well as the optimization of those resources based on the negotiated QoS scheme. The *Binding and Usage* maps a virtual resource to a physical one (i.e maps resources to request), reserves the allocated resources, and triggers the monitoring process. Since in a NVE multiple virtual layers can be built on top of physical one, we design same components to be used at each virtual layer. However, the type of resource/services being offered is different. At a the first and second virtual layer, the *Service Description and Publication* is responsible for describing network services and publishing their information to the broker. In order to get the list of resources of interest, the *Resource Discovery and Selection* interacts with the broker on a request-response basis, and performs another stage of resource selection involving some local criteria/constraints. While the *Service Composition* enables to combine two or more services into a composite service. The *Service Deployment and Test* coordinates the steps involved in service deployment and performs some tests to validate the virtual-to-physical mapping. The *Service Monitoring* monitors the status of deployed services to ensure QoS. Finally, the Negotiation Engine conducts the negotiation of resources with one or more PIPs.

3.6 Case Study

The sequence diagram shown in Figure 8 illustrates the virtual network provisioning process within the proposed architecture. This process consists of a set of interactions between the involved roes, namely: a PIP who is managing the physical infrastructure that is available on-demand; a VIP who discovers and leases virtual resources to offer a platform with customized

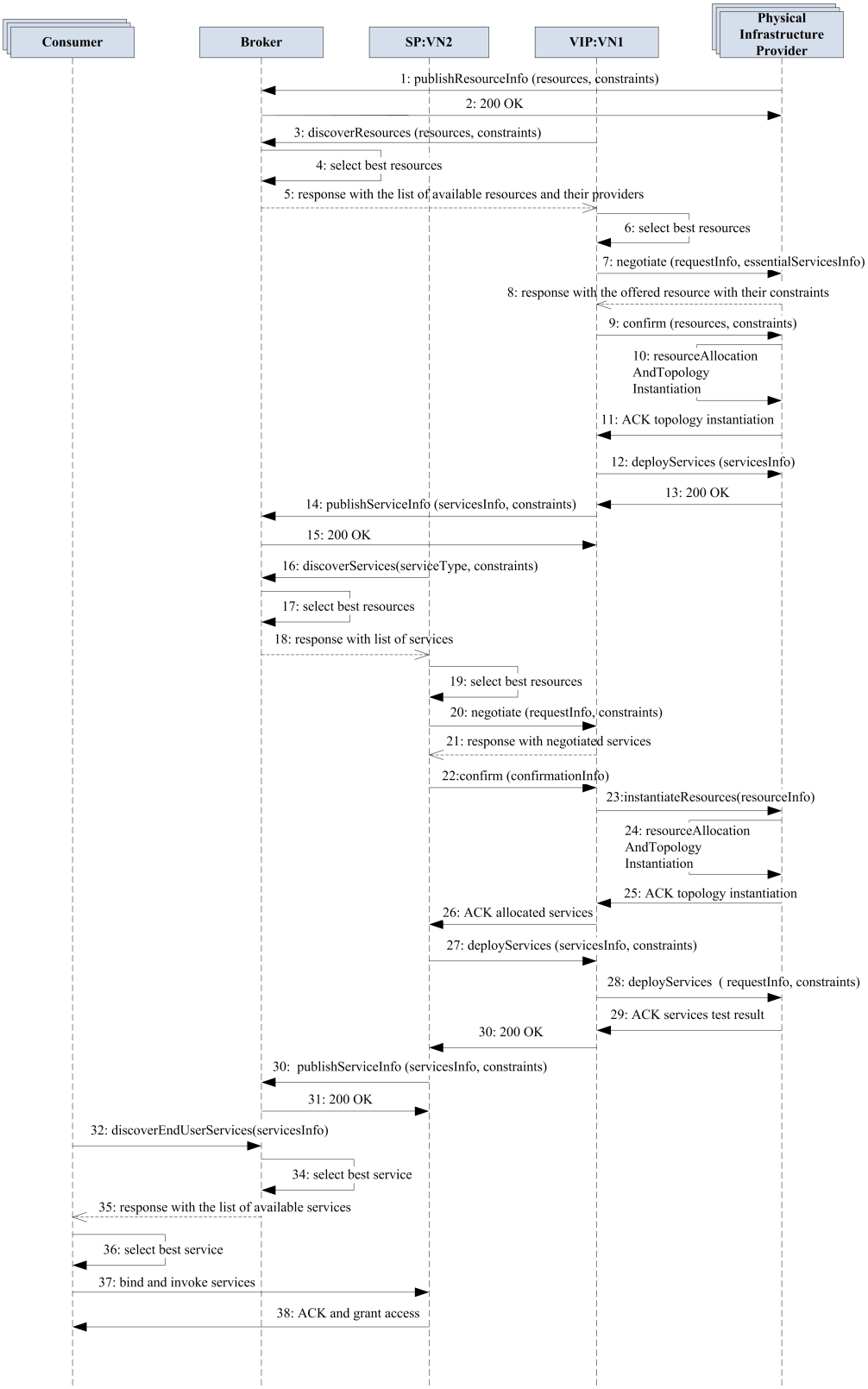

Figure 8: Roles interactions during virtual network provisioning process within the proposed framework

services (such as security and QoS-enabled, enhanced routing services, etc.); a SP who offers the end-user service (the content distribution service) as value added service to customers.

In this scenario, a PIP initiates the process and publishes resource description and constraints related to the offered resources into the Broker using a POST request (step 1). The broker replies back with a confirmation message indicating the result of the operation (200 OK, step 2). In turn, the VIP asks the broker to provide him with the needed resources. He sends a discovery request (GET message, step 3) containing the description of resources, their desired availability, cost, and constraints. Upon receiving a discovery request, the broker executes a selection/matching algorithm to select the optimal resources that comply with the VIP's request. The set of selected resources is sent back to the VIP (step 4). Upon receiving the PIPs list, the best PIP is selected by the VIP, using a selection/matching algorithm (step 5). The VIP then sends a resource negotiation request (step 6), specifying the requested essential services and their constraints, to the selected PIP. The latter replies with a resource negotiation response (step 7), specifying the offered resources and accepted constraints to the VIP, which concludes the negotiation process with a resource negotiation acknowledgment (step 8) confirming the negotiated resources and constraints. At this stage, the PIP carries a resource allocation and virtual topology instantiation process for VN1 (step 9), and sends an acknowledgment (step 10) of the topology instantiation to the VIP. Afterward, the VIP asks the PIP to deploy and test the specified service enablers (step 11), and gets a 200 OK message as reply (step 12). Once the service enablers are deployed and tested, the VIP asks the broker to publish a description of the service enablers and their constraints (step 13), which in case of success results in a 200 OK message (step 14). Meanwhile, a SP (wishing to create VN2) sends the broker a VIP discovery request (step 15) containing a document describing the service enablers to be used, their desired availability, cost, and constraints. The broker replies with a list of VIPs offering service enablers that comply with the request (step 16). Later, in step 17, the SP selects the best VIP to which he submits a service negotiation request (step 18). In steps 19 to 28, interactions related to service enablers' usage negotiation, VN2 topology instantiation, and the deployment of the

content distribution end user service offered by the SP are carried, similarly to the VIP::VN1 case. The main difference lays in the message parameters that refer to a different type of service in this case. When the end user service is successfully deployed and tested, the SP sends its description to the broker. This description is then discovered 4781 by the consumer that uses it to select the best SP. Afterward, the consumer submits a bind and invoke service request to the chosen SP, which in response sends an acknowledgment and grants access to the consumer. The latter then carries the rest of the interactions related to the end user service invocation and usage (those interactions are not shown in the figure).

3.7 Summary

The overall system availability and resource information consistency as well as security are critical in a NVE context. Although we considered a centralized approach in our design process, many techniques can be used to overcome the disadvantages related to centralized systems. For instance, redundancy and load balancing techniques can be used to cope with the problem of single point of failure. On the other hand, adopting decentralized approach, such as structured or unstructured peer-to-peer (P2P) solution is a potential option. Systems based on P2P architecture based are known for their increased scalability since additional node can be added without affecting the availability of the overall systems. However, nodes availability in such type of system is questionable as nodes can join and leave the network with prior notification. In this chapter, we have presented a framework for dynamic publication and discovery of resources in NVE and defined a broker-based architecture that uses the service-oriented business model which was previously presented in Section 2.1. Then, we discussed a case study illustrating the steps required to provision a virtual networks within the proposed framework. Finally, we discussed the different approaches that have been proposed and used for resource discovery and selection in NVE.

Chapter 4

Information Model

In the previous chapter, we established the architecture of the framework we are proposing for resource publication and discovery in NVE, and we gave a detailed description of the responsibility of each component. Moreover, we discussed a use case to show the interactions between roles and the information they exchange.
In this chapter, we present an information model for resource and service description as part of the framework we are proposing. First, we discuss the motivation behind the need of such model. Then, we discuss the work related to information modeling in general and virtual network in particular. Finally, we present the proposed information model and detail the entities supported and their relationships.

4.1 Introduction

Virtualization has been adopted as an effective solution for sharing resources that are used upon request. Network vitalization is the result of applying virtualization in network context. Through virtualization, a physical resource is sliced into many virtual resources with the same or different computational attributes (properties). However, in an environment where multiple infrastructure providers offer heterogeneous resources, each resource has to be described and somehow differentiated. This allows the resource consumer to select the

resources capable of providing a specific service and choose from a wide variety of resources. Moreover, resources can be combined from two or more providers forming a virtual network spanning across multiple administrative domains. In this work, network resources refer the equipment needed to create a network such as a router and switch. Currently, there is no standardized language for accurately describing resources in network virtualization environment where multiple infrastructure providers offer heterogeneous resources and services.

To facilitate the selection and allocation of such resources, each PIP needs to describe the resources and their characteristics in terms of processing power and capabilities. The goal of this information is to identify the resources and their properties. Such an information model is used to model entities at a conceptual level. In fact, as resource provider, a PIP faces significant challenges in describing and advertising the resources it offers. While attempting to maximize the selection likelihood, the offered resources need to be described in fine-grained manner. When a VIP needs to instantiate a virtual network, it first defines a set of requirements that the expected network should meet. The requirements mainly consist of the resource capacity in terms of processing power and network-related capabilities such as bandwidth, delay, etc.

Since a virtual network and resources are provisioned on demand, a common information model to manage and organize information about resources is needed in network virtualization environment. This model would be used by different roles to send request for creating, destroying, or updating resources. Hence, all roles should agree on a single and common data representation, which enables them to efficiently, and reliably exchange information and receive notification (e.g virtual network description, modification request, network-related notifications).

A good analogy for the need to describe resources and services in NVE is the requirements for labelling products and goods that are often provided with a summary description (called labels) that assists the buyers in the purchasing process [55]. A buyer uses the information provided in product labels to compare the characteristics of a given product with the ones

of similar or substitute products. Driven by many constraints and preferences (e.g quality of the product, reputation, price, ingredients, warranty), such comparison assists the buyer in evaluating a product and making a decision about selection the product in question or not. The work presented in [56] attempts to clarify what needs to be included when describing services' properties. Because "electronic" services are consumed in a "semi-automated" manner, the author argues that the description of a service should answer some service related questions such as: "how you determine how to request a service? What is the identity of the service provider? Where and when the service is available? What quality of service can be guaranteed? Does the provider offer any type of discounts?" Moreover, an accurate description of a service should include the related functionality and the associated constraints.

In fact, from a resource requestor perspective in NVE, a resource description should be accurate and provide the requestors with the necessary details (including resource's functionality and constraints). Formulating resource requests and modeling such resources is crucial but yet complex. New and well-defined schema is needed for handling automated and on-demand provisioning of resources and to cope with the challenge of selecting appropriate resources based on their advertised description. Therefore, roles have to use the same model to be able to express their needs; and have a common understanding related to the exchanged information. Consequently, role-to-role interfaces (or information carriers) are key elements for successful communication in the network visualization architecture through which roles communicate their needs, and receive notifications, related to virtual resources they are concerned about. These interfaces, for instance, enable the VIP to send a virtual network instantiation request to a PIP along with the characteristics and constraint on the desired virtual network. Moreover, these interfaces enable two different roles negotiate requests and QoS related concerns.

In this work, we are only concerned about the two following types of virtual networks: (1) a virtual network made of virtual routers and virtual switches. (2) A network of interconnected virtual nodes providing computing or storage resources (e.g interconnected

virtual machine in Cloud Computing, Data Centers).

4.2 Related Work to Resource Description

In this section, we survey the previous work on resource description in networking and computing in general. We then discuss the work related to resource modeling in physical network in general and virtual network in detail.

The work related to resource description in distributed environments is multifold. Many resource description languages and specifications have been proposed. Table 1 shows the existing description languages grouped by network/computing area.

Table 1: Summary of some existing description languages grouped by networking/computing area

Networking/ Computing area	Description Language
Grid computing	VXDL [57], vgDL [58] SWORD [59], GLUE [60]
Cloud Computing	CloudML [61], Data Centers Markup Language DCML [62], CloudView [63]
Network resources	NDL [64], NRDL [65] , NML [66] NevML [67], CIM [68], cNIS [69], DEN-ng [70]
Virtual Networks	Houidi et al. [16], VXDL [57], VN-SLA [55], NNDL [71]
WSDL-based schema	INDL [72], RSpec [73] (GENI) WSDL-based schema [53], IETF VNMIM [74] (work in progress), FleRD [75], NOVI [76]
Services description and semantic Web	USDL [77], WSDL, WADL [78], OWL-S [79], RDF [80]

After analyzing the aforementioned specifications, we conclude that many network-centric information models/languages have been proposed to model physical network-related

aspects. While the main challenge in modeling virtual networks is to efficiently describe virtual resources (along with their functional and non-functional attributes), little work have been initiated in this area. At the time of writing, there are no standardized models for virtual networks. To give a literature review on virtual networks, we consider only the most relevant and recent works that address different aspects related to network virtualization environment.

Based on the Resource Description Framework (RDF) [80] and Semantic Web, the Network Description Language (NDL) [64] has mainly been introduced to model hybrid networks. Consequently, RDF vocabulary is used to describe network-related concepts/objects in NDL, which results in what so-called RDF documents. NDL is an information model that is designed in a modular manner. It encompasses a group of five independent schemas, namely, topology, layer, capability, domain, and physical schema. Thus, it allows describing network physical-related aspects such as network topology, network devices and their capabilities, interfaces and connections between them as well as describing administrative domains and generic properties of network technologies. NDL has been used [76] and extended in many projects such as GENI project [81]. Even though it was widely accepted and recommended by W3C, NDL does not offer support for describing virtual network aspects nor specifying constraints and QoS specifications.

Mainly designed for virtual grid applications, Virtual Resource and Interconnection Networks Description Language (VXDL) [57] is a language for virtual resources interconnection networks specification and modeling. The authors define a virtual infrastructure as an aggregated set of interconnected virtual resources. VXDL allows describing all components of a virtual infrastructure including their network topology. Thus, VXDL introduced the notion of "timeline" or period of resource utilization and resources can be described individually or in groups. A typical VXDL document comprises the following: general resource description, resources description, network topology description, and timeline description.

On the other hand, Virtual Network-Service Level Agreement (VN-SLA) [55] provides a

preliminary schema for defining a service level agreement for virtual networks. Composed of three sections, it allows defining: (1) various actors (InP, VNP, Client); (2) virtual resources properties and their relationships as well as the Service Level Specification (SLS); (3) finally, the obligations which formulates each actor's responsibilities and the consequences of inability to meet the specified service levels. Considering virtual recourses as services with minimum granularity, authors in [53] extend the WSDL and propose a WSDL-based model for virtual network resource description. The main goal of this model is to support the dynamic update of resource information. In this model, any resource description defines the nature of the resources being offered and the endpoint (location) determining where these resources can be accessed.

The Flexible Resource Description Language (FLeRD) [75] is proposed for multiple-provider virtual network architectures, more specifically, virtual networks connecting cloud resources (or CloudNets). It emphasizes flexibility, vagueness, white and black listing of properties as well as describing resources while allowing the possibility to omit some resource specifications. Another interesting point of FLeRD is the support for the formulation of the mapping between virtual and physical resources.

As a result of in integrating NDL and Cinegrid Description Language (CDL) (a service-oriented modeling language), the Infrastructure and Network Description Language (INDL) aims at providing technology independent descriptions of computing infrastructures. Based on ontologies and Semantic Web approaches, INDL's main goal is to decouple connectivity, functionality and virtualization of resources so that flexibility is ensured and new types can be added without affecting existing schema. In INDL, network resources are represented by the Resource class and its subclasses (such as Node, Network Element, and Node Component). While network services (i.e Storage Service, Stream Service, etc.) are represented as subclasses of the Service class.

As work in progress, a virtual network management information model is proposed in [74] for mainly managing virtual networks in data centers. In this model, virtual nodes that are instantiated on the same physical node are defined in a group of virtual nodes and

the mapping to physical ones is expressed as well. Consisting of the starting point of our work, Houidi et al. [17] proposed a schema for automated virtual networks provisioning whose objective is to define the properties of virtual resources and their relationships. In such a schema, a network element is considered as the basic building component and has functional and non-functional attributes. Functional attributes represents characteristics, properties and functions of a network element. Whereas non-functional attributes specify criteria and constraints (such a location, cost, QoS). Even though it is intended to define virtual resources, this schema does not take into consideration dynamic resource information update nor covers all the aspects presented in this thesis such as describing virtual network as a whole, mapping between virtual and physical resources, describing network services, and formulating the relationships between roles (actors/parties) and resources/services.

Each of the aforementioned languages address specific domain and do not cover all the aspects related to virtual networks. In the following, we highlight the most interesting point of each aforementioned solutions and we conclude how they differ from the proposed model.

Although NDL supports various network aspects and has been extended in many works, it lacks support for virtual network concepts. The WSDL-based model has an interesting point in supporting dynamic resource update but does not provide support for detailed virtual resources nor network services modeling. VN-SLA allows the modeling of virtual resources; however, it does not support network services. Thus, FLeRD focuses on the formulation of mapping (virtual-to-physical layers) of network elements while no support for modeling network topology and roles is provided. VXDL allows describing individual resources in detail but does not model virtual network aspects such as roles and network services. INDL does support network services and resources whereas virtual-to-physical mapping is not considered. None of these solutions provide detailed functional and non-functional attributes related to resources and network services, nor do they model the interactions among roles, resources and services. Besides, the proposed model formulates a virtual network layer including all the involved roles, services and resources as a whole.

Table 2: Comparison of the existing information model with our requirements

	Fine-grained description of virtual resources	Virtual to physical mapping information	Formality and expressiveness	Virtual network topology mapping	Support for formulation network services	Support for aggregation of network services
Houidi et al. [16]	–	X	X	–	–	–
VN-SLA [55]	–	–	X	–	X	–
NNDL [71]	X	–	X	–	–	–
VXDL [57]	X	–	X	–	–	–
INDL [72]	X	–	X	–	X	–
WSDL extension [53]	–	–	X	–	–	–
FLeRD [75]	X	X	X	X	–	–
Our model [9]	X	X	X	X	X	X

4.3 Requirements for an Information Model in Network Virualization Environment

Among the several requirements that have led to the design of our proposed information model. Reference [82] introduces a process to develop an information model. The author argues that a "quality" information model should have the following characteristics: "sharable, stable, extensible, well-structured, precise and unambiguous". In addition to these characteristics, an information model should cope with the complexity that the virtual network provisioning process raises in terms of resource description, management and organization. To be suitable for NVE, we believe that an information model should meet the following requirements:

- **Fine-grained description of virtual resources.** The proposed model should clearly enable the description of the functional and non-functional attributes in detail as well as their related constraints and QoS scheme.
- **Mapping information.** Thanks to virtualization technology, a physical device (i.e server, router) can host many virtual devices. The proposed model should enable to represent the mapping of virtual resources to physical resources i.e on which physical resource a virtual resource is created.
- **Formality and expressiveness.** A model could be seen as a template that one has to follow to represent a given object. Such a model should be formal and should provide the structure of information used to describe resources/services with an acceptable degree of expressiveness.
- **Enable inter-role communication.** The model should seamlessly enable data exchanges between heterogeneous roles/services.
- **Interoperability between roles.** The proposed model should effectively enable information sharing and data exchanges between different NVE roles without ambiguity.

- **Virtual network topology mapping.** A VNet has a virtual network topology (VNT) which is a subset of a physical network topology (PNT). The proposed model should enable the representation of VNT as well as its corresponding mapping to the PNT.

- **Connection information between resources, services and roles.** The proposed model should be able to represent all the connections between resources, roles and services and the relationships they might have.

- **Extensibility and flexibility.** The model should provide a solution to flexibly describe network resources and services. Thus, any potential extension should be taken into consideration. Hence, it should be easy to extend the model in order to represent new entities.

- **Support for aggregation of network services.** A service can be composed by aggregating two or more services. For instance, a SP can offer composite services by combining many service building blocks.

- **Support for formulation network services.** The model should not be limited to modeling virtual resources and should allow for modeling a variety network services.

- **Information representation and portability.** The proposed model should not have any implementation-related constraints. Any platform-independent language that ensures interoperability could be used to implement such a model.

4.4 The Proposed Information Model

4.4.1 High-level Overview

Figure 9 gives a high-level overview of the proposed model. Our information model revolves around modeling three main concepts and their relationships: roles; services; and resources [9]. Roles are business organizations that collaborate to offer/consume resources

and services and exchange information related to these resources/services. A role can be a resource provider offering and managing virtualized resources, and at the same time can be a resource consumer accessing virtualized resources/services. In addition, a role can act as service provider offering and managing network services, or as a service consumer subscribing to network services. In our model, network resources are mapped onto network services (i.e. network resources are considered as low level network services). Finally, just like web services, various levels of network services can be published, dynamically discovered, composed, and used, in our model.

We take into consideration the different roles and their relationships to physical/virtual topologies and various levels of services . A *TargetedNetwork* can be physical or virtual network. To describe a virtual network, we consider a *TargetedNetwork* to be the base entity as well as the root element of all instantiated description documents. A *TargetedNetwork* can be composed of one or many virtual networks and one or many physical networks. A PhysicalNetwork has a *PhysicalNetworkTopology* and is composed of a set of physical nodes connected by physical links. A *VirtualNetwork* has a *VirtualNetworkTopology*, which is a subset of the underlying physical topology. A virtual network topology can be composed of one or multiple virtual ones, thus forming a hierarchy. A virtual network is composed of a set of *VirtualNodes*, each node having one or many *VirtualInterfaces* and being connected to another virtual node by a *VirtualLink*. Virtual nodes that are instantiated on the same physical device are grouped in a *VirtualNodeGroup* that is mapped to a physical node. Although we are not concerned about modeling physical networks related entities, we only model a physical network as a set of *PhysicalNodes* where a given group of virtual nodes is mapped. The different roles and their interactions with different entities are modeled as follows: (1) A *PhysicalInfrProvider* (PIP) owns and operates a *PhysicalNetwork*; offers *EssentialServices*; and instantiates one or multiple *VirtualNetworks*; (2) A *VirtualInfrProvider* (VIP) manages and operates *VirtualNetworks* and offers *ServiceEnablers*; (3) A *ServiceProvider* (SP) manages and operates *VirtualNetworks* and offers *ServiceBuildingBlocks* and *EndUserServices*. An end user service can be created by

combining one or more service building block services; and (4) Considered as end-user, a Consumer subscribes to/uses one or multiple EndUserServices that are accessible via *PhysicalNetworks* and *VirtualNetworks*.

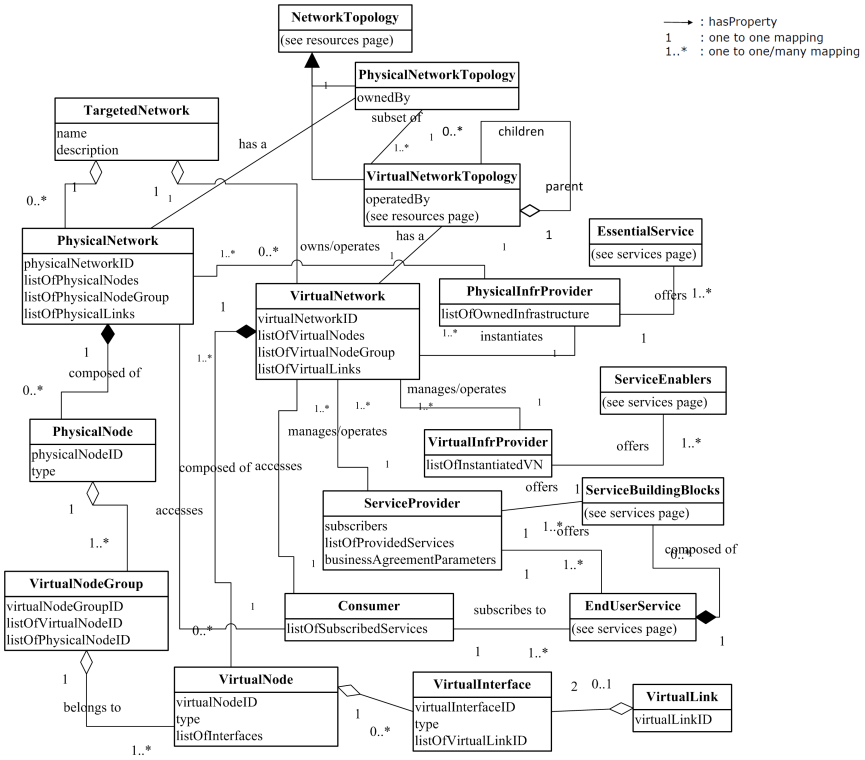

Figure 9: High-level overview of the proposed information model

4.4.2 Detailed Description

Figure 10 and Figure 11 respectively show the resource level view and the service level view. In the resource level view, we consider a *NetworkElement* (NE) as the basic building component of a virtual network that can be a Node, Link, Interface, or Path. A NE has a name, availability, start time that specifies when the resource is available, and a period that determines for how long the resource is available. The status attributes represent NE's state (available, allocated, etc.). Since a NE can span across multiples domains, a NE belongs to a *NetworkDomain*, which in turn has an *AdministrativeDomain*.

A Node can be either a *PhysicalNode* or *VirtualNode*. Represented in the class Node, a

node has a *RoutingPlatform* and *GeoLocation*, and encompasses common attributes needed for describing a network node, namely, a network stack, a type (i.e virtual switch, virtual router, virtual machine, etc) and an IP address. Besides attributes such as the vendor, model, and substrate node group, a physical node may aggregate virtual nodes and interfaces, whereas a *VirtualNode* (VN) is uniquely identified; and has an initial and maximum capacity in terms of computational capabilities. Each VN aggregates one or multiple virtual interfaces. An Interface represents a physical/virtual network interface controller (NIC); and has a type (i.e Ethernet, radio), rate and MAC address. Depending on its capacity, a physical link can be divided into slices using virtualization techniques (i.e ATM, MPLS) to support one or multiple virtual links. A Link has characteristics such as minimal delay, type, bandwidth, throughput, good-put and type of connectivity; and an end point that determines the source node and destination node. Each *VirtualLink* has a tag, and initial and maximum allocated bandwidth. Virtual interfaces are connected by a virtual link. A *PhysicalLink* has a limited number of supported virtual links and an additional attribute for defining available bandwidth. A *Path* represents a set of links, starts at *beginNode* and ends at *endNode*.

To represent nodes' functional and non-functional characteristics, a node has an association with the following two entities: (1) Node Functional Parameters: consists of characteristics/properties related to the functioning of a node such as operating system type, software version, and the type of the network management system. It is composed of: (a) Storage parameters which determine the available disk space, storage type, and number of storage units; (b) memory parameters which represent the size, capacity, and type of the available memory; and (c) CPU parameters which represent the information about the available processing unit(s). (2) Node Non-Functional Parameters: this class defines constraints, QoS scheme, and desired criteria that should be met when selecting a resource, namely: cost, rank, and percentage of failure. In turn, non-functional attributes are composed of the following: (a) Performance parameters representing node performance properties such as response time, uptime, capacity, and reliability level. (b) Security level parameters defining security properties that a node supports like hashing techniques (i.e

Checksums, cryptographic hash functions), encryption methods (i.e symmetric, asymmetric) and security properties (i.e confidentiality, integrity). (c) QoS parameters representing QoS related characteristics including the average packet loss, jitter, delay, and bit rate. (d) Input and output parameters representing properties used to monitor node's workload status (e.g CPU and memory state, I/O devices, etc.).

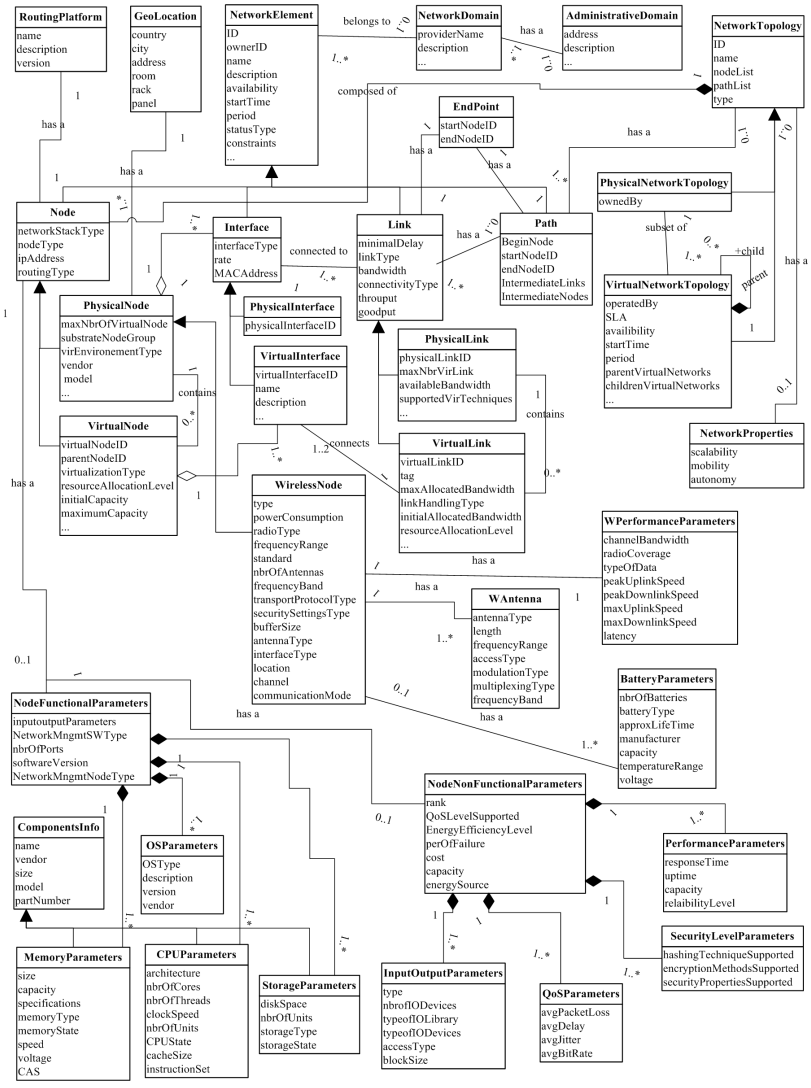

Figure 10: Resource view of the proposed information model

We model network topology as physical/virtual topology. In general, a network topology has a name, type (i.e bus, ring), path list, and is composed of a set of nodes. Moreover, a network topology has *NetworkProperties* that is a set of characteristics applied to wireless and

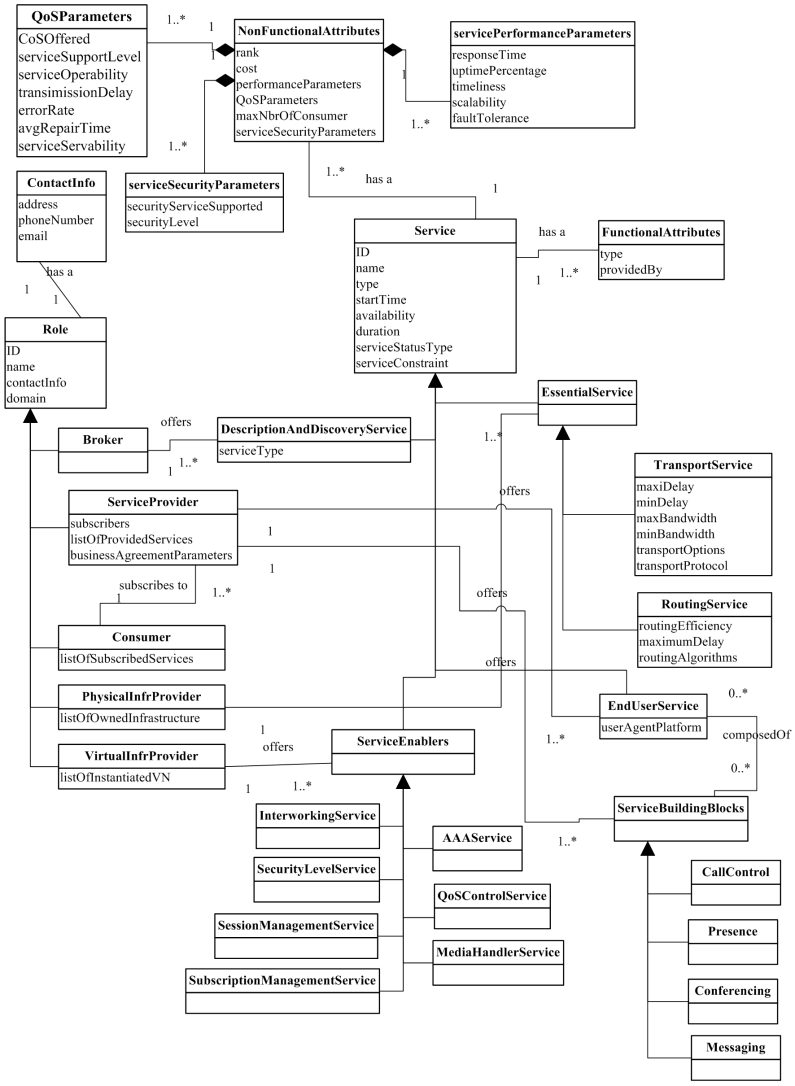

Figure 11: Service view of the proposed information model

wireless sensor networks (i.e scalability, mobility and autonomy). Representing the topology of a virtual network, a virtual topology is a subset of a physical one and can be hierarchical so that a virtual topology can be instantiated on top of one or multiple virtual topologies. Thus, this leads to have hierarchical associations among virtual networks. Besides, it contains attributes related to availability, start time, period, and a reference to its operator. In the service level view shown in Figure 11, a role represents an organization, identified by a name or id and has contact information. Different roles are modeled as follows: (1) broker represents the resource and service repository; (2) Service provider represents a SP; (3) consumer represents an end user which subscribes to services offered by a SP; (4) Physical infrastructure provider represents a PIP; (5) Virtual infrastructure provider represents a VIP. Each role is associated with a service entity which indicates the type of service it offers.

Just like NE, a service represents the base class for describing services. A service has the following sub-classes: (1) description and discovery service offered by the broker and representing services needed for publishing and discovering resources/services; (2) Essential service are transport service and routing service; (3) End-user service representing services destined to end users and composed of one or many service building blocks namely call control, presence, conferencing, and messaging; and (4) Service enablers defining the support functions needed for the operation of end user services. Service enablers comprise the following: Interworking, security level, session management, subscription management, AAA service, QoS control, media handler. Each service is associated with functional attributes as well as non-functional attributes. We divide the latter into three categories: (1) QoS defining characteristics such as the offered class of service, support level, error rate, average repair time, and transmission delay; (2) Service performance representing properties that are related to service performance, namely, scalability and fault tolerance, response time, and uptime percentage, etc.; and (3) service security defining the security service and the level supported. Furthermore, common properties like the rank of a service, cost, and maximum number of supported users can be expressed as well.

A *WirelessNode* is a *PhysicalNode* and can contain one or more virtual nodes.

The *WirelessNode* class describes the characteristics of a wireless node (such as power consumption, radio type, frequency range, communication mode, etc). Additionally, a wireless node has an antenna, battery and performance parameters.

The class *WAntenna* defines the set of properties that an antenna can have (such as antenna type, frequency range, and modulation type). Additionally, the *WPerformanceParameters* class represents the different type of properties that are related to the performance of a wireless node including channel bandwidth, radio coverage and latency as well as maximum down/up link speed and type of data supported.
Moreover, we defined a set of enumeration types which are listed in Appendix A. Such enumeration types are used to formulate various entities' attributes. Figure 34 in Section A.1 shows the enumerations related to a *node* entity. For instance, such enums specify the type of a node (virtual router, virtual switch, etc.), the virtual environment used (e.g., XEN or VMWare, etc.) and so on. The figure in Section A.2 shows enumerations types related to network links. Whereas Sections A.3, A.4 and A.5 define enumerations types related to network services, security and wireless node respectively.

4.5 Summary

In network virtualization environment, multiple providers provide heterogeneous resources and services which results in a large pool of virtual resources that can be discovered and consumed on demand. Thus, virtual networks are created by aggregating resources that span across different domains. Prior to instantiate a virtual network, a virtual infrastructure provider would need to discover and select the resource of interest based on their description (their characteristics in terms of processing power). Consequently, an information model is needed to assist physical infrastructure providers in describing the resources they offer and VIPs in selecting the best resources that match their requirements. This chapter presented a multi-service and multi-role [9] integrated information model for describing resources and services in NVE. Such model enables the description of physical and virtual resources as well as network services. We have first defined a set of requirements we believe an information

model should meet. Finally, we surveyed and present the existing information models that have been proposed to model network resources in general and the ones proposed for NVEin particular.

Chapter 5

Design and Implementation

In this chapter, we present the design and development of proof of concept prototype. We first introduce the requirements for the implementation in Section 5.2, then we describe the software architecture of the implementation in Section 5.3 along with the developed components and graphical user interfaces, and finally we detail the expanded use case scenario as a case study in Section 5.5. In the next chapter, the hardware used and the prototype setup will be presented.

5.1 Overview

We presented the architecture of the proposed framework for resource publication and discovery in NVE and discussed its components in Chapter 3. Then, in Chapter 4 we introduced an information model for describing various resources and formulating different kind of requests. We use both the proposed architecture and the information model to implement a proof of concept prototype to demonstrate the feasibility of the proposed framework. However, due to time constraint, we mainly focus on the operations enabling the publication, discovery and negotiation of resources as well as the automated instantiation of virtual topology as described in virtual network request. Therefore, we implemented only a subset of the overall architecture. However, for simplicity reasons, we combine the VIP

and SP roles. As matter of fact, implementing the activities related to service providers (e.g., deploying end-to-end services) is out of the scope of this thesis and is not taken into consideration. As a concrete use case, we selected the secure content distribution scenario presented in Chapter 3 to demonstrate the interactions taken place between the roles.

5.2 Requirements for the Implementation

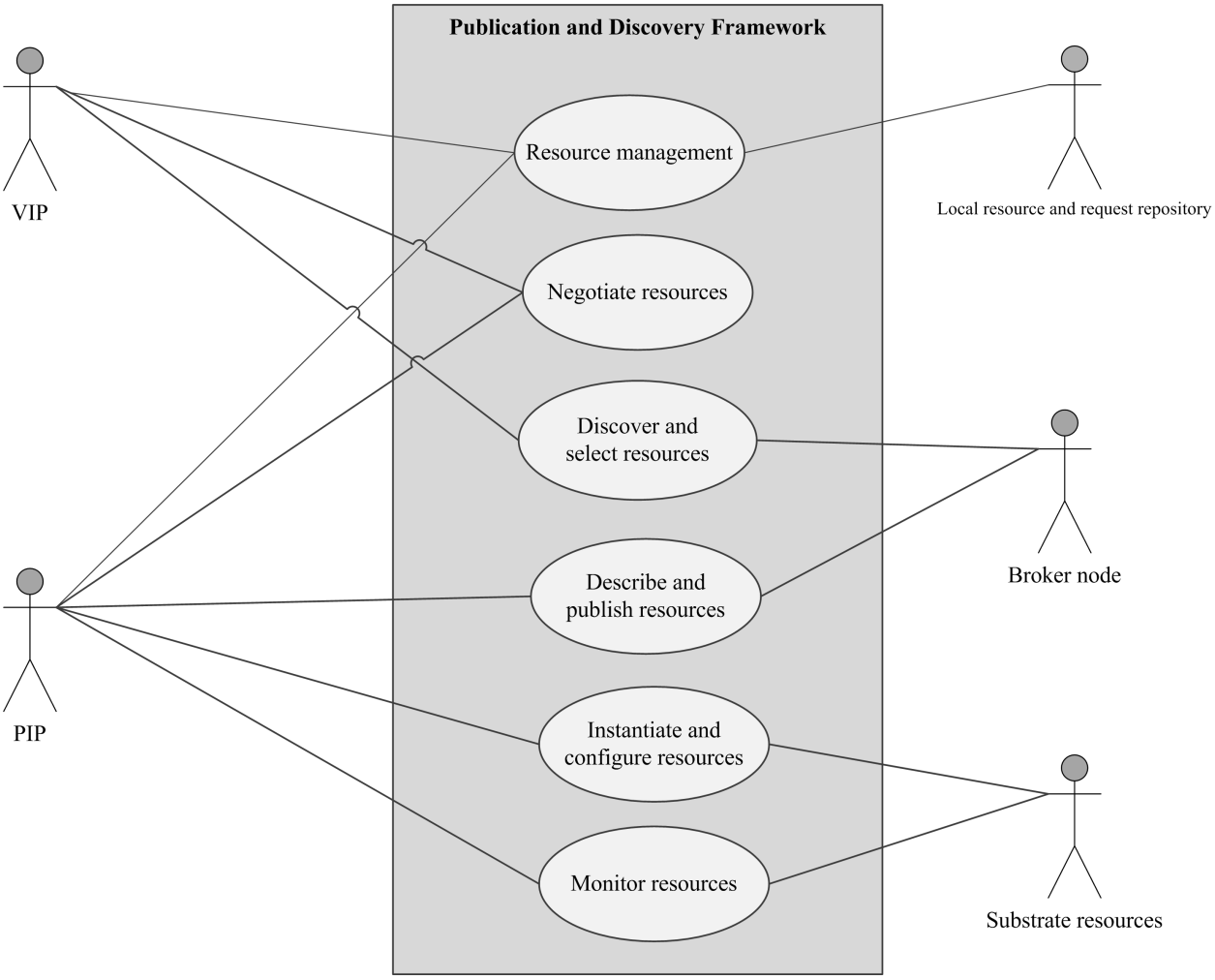

Figure 12: Overall framework use case

The purpose of this of the work presented in this chapter is to put in practice the framework we have established in previous chapters and provide a working solution for

publication, discovery, negotiation and instantiation of virtual resources. Such solution should meet the following requirements:

- **Resource management** the prototype should enable PIPs and VIPs to manage the information about resources stored in the local resource repository that each role disposes.
- **Interaction between roles** interactions between roles is the key requirement that should the implemented first. A mechanism that enables various roles to exchange information about resources should be put in place. This should enable a role to send requests and receive a reply to each request sent.
- **Resource description and publication** the implemented prototype should allow various PIPs to describe physical and virtual resources and publish (register) the information about resources they offer into a public repository (i.e., resource broker) through public interfaces.
- **Resource discovery** the prototype should allow various VIPs to send resource discovery requests to the resource broker to discover and select virtual resources of interest. To be processed successfully, resource discovery requests should include the properties of the resources needed as well as selection constraints (as detailed in Section 5.4.6.1).
- **Resource selection** the resource selection algorithm to be used should be efficient and accurate in processing discovery requests and take into account the constraints and requirements specified in each request.
- **Resource negotiation** the solution should enable multiple VIPs to negotiate the selected resource with the appropriate PIPs through public interfaces. In the negotiation process, roles should be able to modify (edit) a request as well as track the received requests. Hence, all the sent and received requests should be stored in a local repository for later consultation.

- **Virtual topology instantiation and management** the prototype should enable the instantiation of virtual topology. This should be done by creating the requested virtual machines and connecting them through virtual links exactly as described in the virtual topology instantiation request. Moreover, the topology instantiation process should be entirely automated meaning that any manual intervention should be eliminated.

- **Information visualization** the prototype should provide a set of graphical user interfaces that would enable roles to visualize the information related to the resources they manage or the sent or received requests. For example, a PIP should be able to visualize the list of the resources he manages along with their status (e.g., published, instantiated, etc.). Moreover, the virtual topology should be displayed as connected graph nodes. This should reflect the created virtual resources and their connections.

In addition, the resource publication and discovery process should be automated and requires minimal human intervention. Figure 12 illustrates the use cases that should be implemented.

5.3 Software Architecture

Figure 13 depicts the software architecture of the implemented prototype and the technologies used. It consists of three main subsystems, each of which, is intended to realize the functionality of the respective role (i.e Broker, PIP, VIP), and encompasses a repository (data store) containing information about resources and services. Moreover, each subsystem exposes a set of its functionality via public interfaces intended to enable communication with other subsystems belonging to other roles. The interfaces consist of RESTful web services.

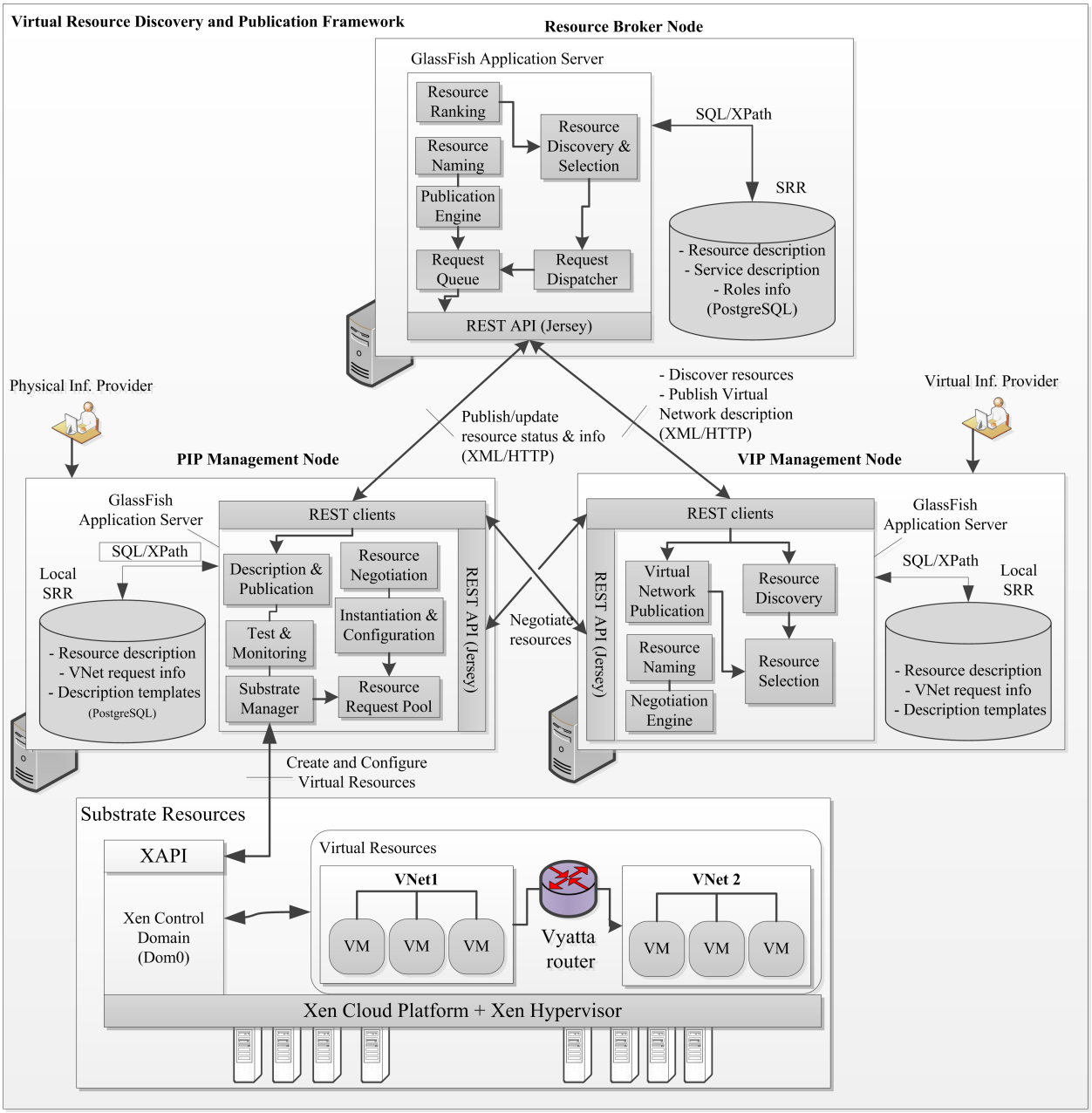

Figure 13: The software architecture of the implemented prototype

5.4 Implementation

Figure 22 depicts the prototype setup. We installed and configured each subsystem on a separate machine, namely: the PIP management node; the resource broker management node; the VIP Management node. As for the substrate resources, we installed XCP on three machines and prepared a set of Vyatta and Ubunutu virtual machines to be used when instantiating the virtual topology. In this section, we present the developed interfaces that enable the interaction with each subsystem presented in Section 5.3 and discuss their functionality.

5.4.1 The Technologies and Tools Used

We used the Java programming language to implement the prototype. The REST web services were implemented using Jersey [83] framework that is an open source JAX-RS (JSR 311) reference implementation . We selected Grizzly web server [84] to deploy the web services. Moreover, we used JAXB 2 [85] for marshaling and un-marshaling the XML data carried in various REST messages' body. We opted to use XML to describe the resources and formulate the various requests exchanged between roles (e.g discovery and negotiation requests). Thus, the XSD (XML-Schema Definitions) was used to define and validate the structure of the data models and specify constraints on the data contained in the XML documents. Consequently, each document exchanged between two subsystems is a data model (i.e an instance of our proposed information model). The various used XSD are listed in Appendix C.

5.4.1.1 Data Sources

We selected the open source RDBMS PostgreSQL [86] which provides native XML support, SQL/XML publishing/querying functions, full-text search as well as full-text indexing and XPath support. Moreover, PostgreSQL stores an XML document in its text representation, which results in fast information retrieval and provided us with more flexibility in describing

resources by eliminating the need to change tables' schema upon the alteration of the structure of the resource's description.

5.4.1.2 Platform Virtualization

As virtualization platform, we selected Xen Cloud Platform (XCP) [87] because it includes the Xen Hypervisor as well as XAPI (Xen Management API) and supports large number of guest operating systems. Other alternatives to virtualize infrastructures include, but not limited to, VMWare vSphere [20], KVM [88], Citrix XenServer [21], Microsoft Hyper-V [19].

The Xen project uses the term domain to refer to any virtual machine created by Xen. As per its design, there are two Xen domain types: Domain0 (Dom0) and DomainU (DomU). Dom0 is the privileged domain and is created at boot time; has direct access to hardware and can be seen as the hypervisor itself: it is responsible for creating and managing the guest machines (DomUs). Whereas, a DomU is unprivileged machine within which a guest operating system is installed and shares the underlying physical resources with other VMs. XAPI is the default toolstacks that Xen provides to manage a virtualized machine. It exposes Dom0's services as XML-RPC services, which enables programmatic access to, and remote administration of, Xen-enabled VMs. Xen supports both full virtualization and para-virtualization, and has demonstrated to be the virtualization platform of choice due to its capabilities in terms of performance, features and isolation level among virtual machines. Therefore, to communicate programmatically with the virtualized physical nodes, we used XenServer's Java SDK to implement the substrate manager module and eliminate the manual management of Xen-enabled hosts. XenServer's SDk is a set of programming interfaces developed by Citrix to provide control over the various virtual machines and physical hosts (e.g., create or clone, power up or off virtual machine, etc). This enabled us to dispatch the necessary VM management and control commands to all virtual machines.

In this section, we present the developed interfaces that enable the interaction with each subsystem presented in Section 5.3 and discuss their functionality.

5.4.2 Resource Publication and Management

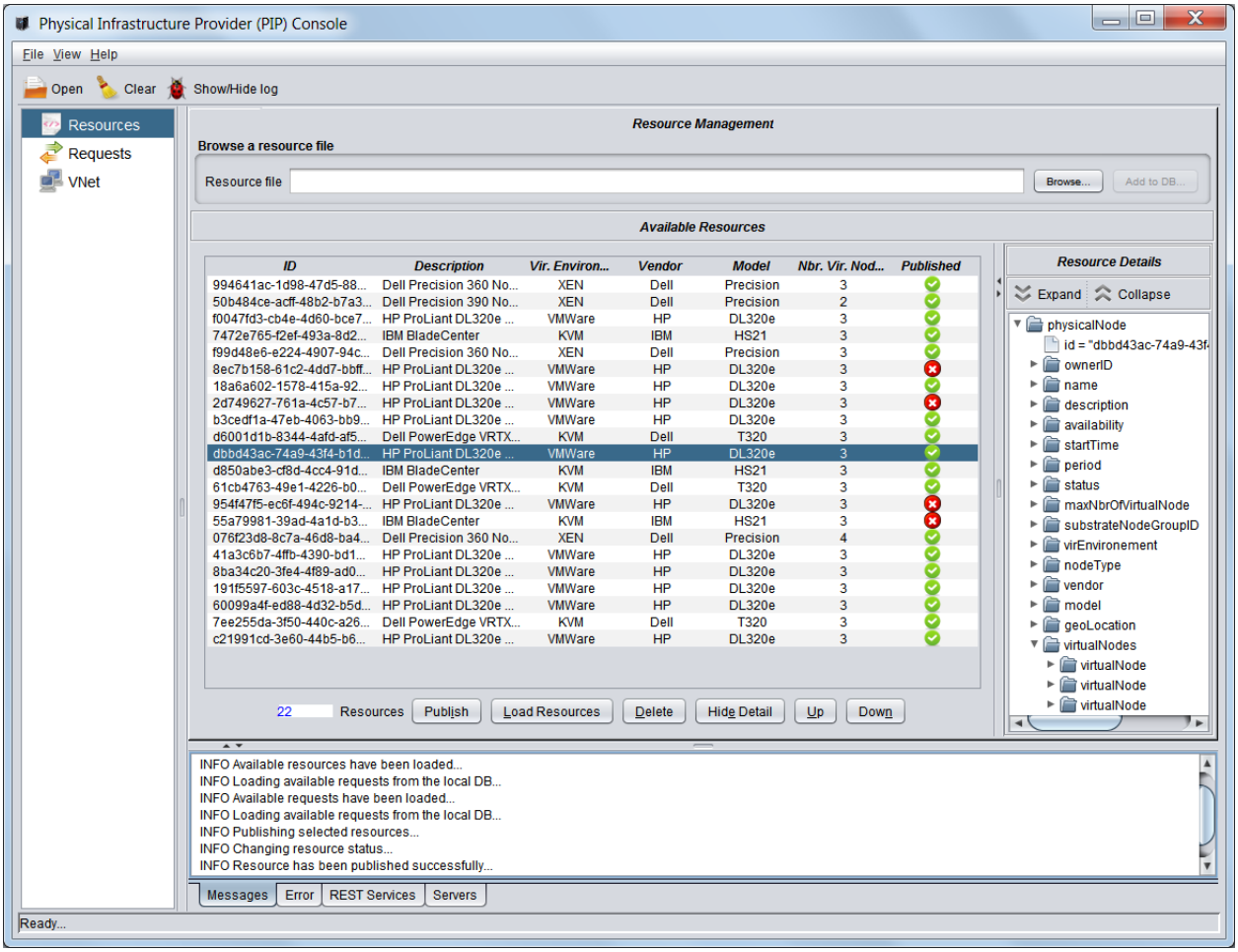

Figure 14: The Physical Infrastructure Provider resource management interface

Figure 14 shows the interface through which a PIP perform CRUD-based operations (create, read, update, delete) to manage the information about the resources he owns. This view enables the PIP to visualize the list of available resources that are contained in the PIP's local repository along with their key information such as a brief description of the resource, the virtual environment, and the status of each resource whether it is published into the broker or not. To add a new resource description to PIP's local repository, the user clicks on the browse button to locate a resource description document (i.e an XML file), and added to the list of resources (by clicking on add to DB button). The resource description is updated if it already exists; otherwise, it is added and marked not published. An example of resource description document is listed in Appendix C, Section C.4. Additionally, at any time, the user can select a resource to visualize its description's details. This is shown in the right panel of the interfaces as a tree view. To publish a resource to the broker, the users selects a resource from the list and clicks on the publish button and the broker's response is displayed afterwards indicating whether it has been published successfully or not. For more details, the message logs of the implemented components' that are involved in the publication process are listed in Appendix D.

5.4.3 Resource Discovery

Figure 15 illustrates the interface used to perform the resource discovery process. It allows the VIP to load a resource discovery request (as shown in Section 5.4.6.1, Listing 5.1) and send it to the broker and visualize the discovered resources using the Discover button which sends the previously loaded discovery document as shown in the right panel. When received from the broker, the list of discovered resources is displayed in the panel located in the center of the interface. From this list, a VIP can browse the list, and perform another selection process by visually analyzing the resources' functional attributes and capabilities. Only the resources of interest are added to the selected resources list and included in the negotiation request that is later sent to the PIP using the Create Request button.

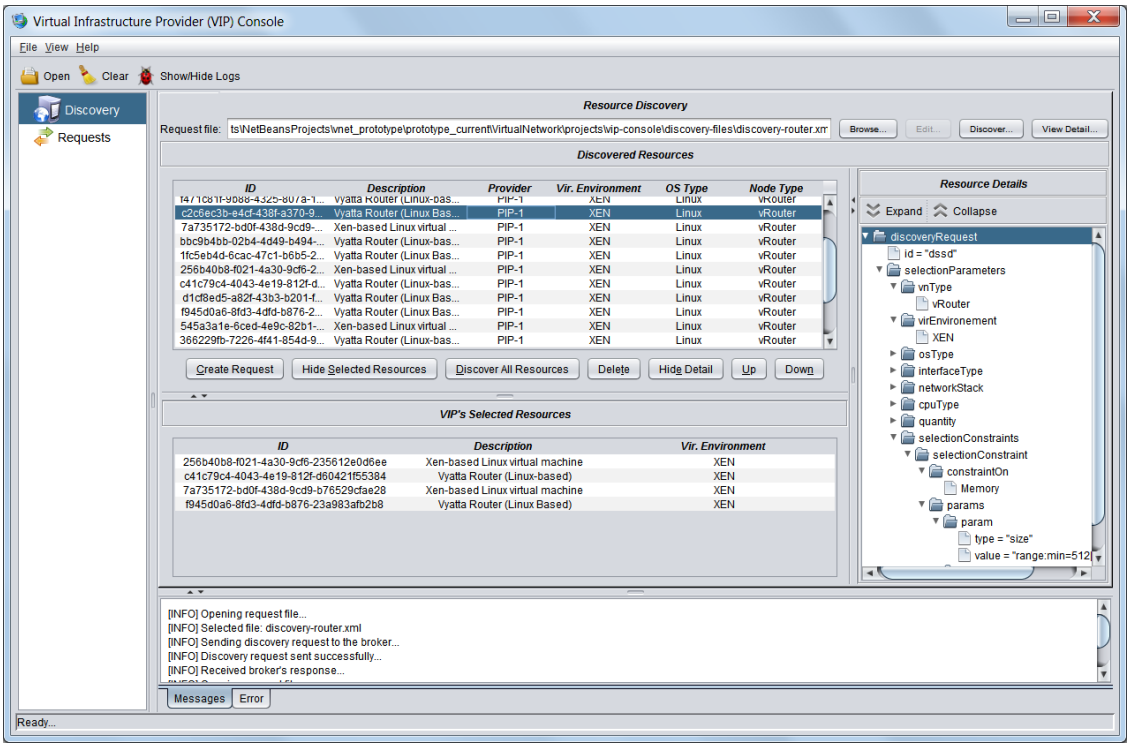

Figure 15: The Virtual Infrastructure Provider discovery interface

5.4.4 Resource Negotiation

In our implementation, resource negotiation takes place between a PIP and a VIP. The VIP triggers this process by sending a negotiation request to the PIP. Besides information about the virtual resources, the request includes, but not limited to, information such as the price wanted per resource, the quantity wanted and additional details formulated as comments. The XSD used for formulating a resource negotiation request is shown in Appendix C, Section C.3. During this process, the web services of both VIP and PIP must be up and running. The request is embedded in a REST POST message and sent to the PIP through its negotiation web service URI. For the roles to keep track of their resource negotiation activities, the negotiation requests are stored in their belonging repository for later consultation. When received, the request is marked not processed (pending) and stored in the repository. Then, a background thread retrieves and adds it to the list of requests that are displayed in the negotiation interfaces. A may reject, or accept a request.

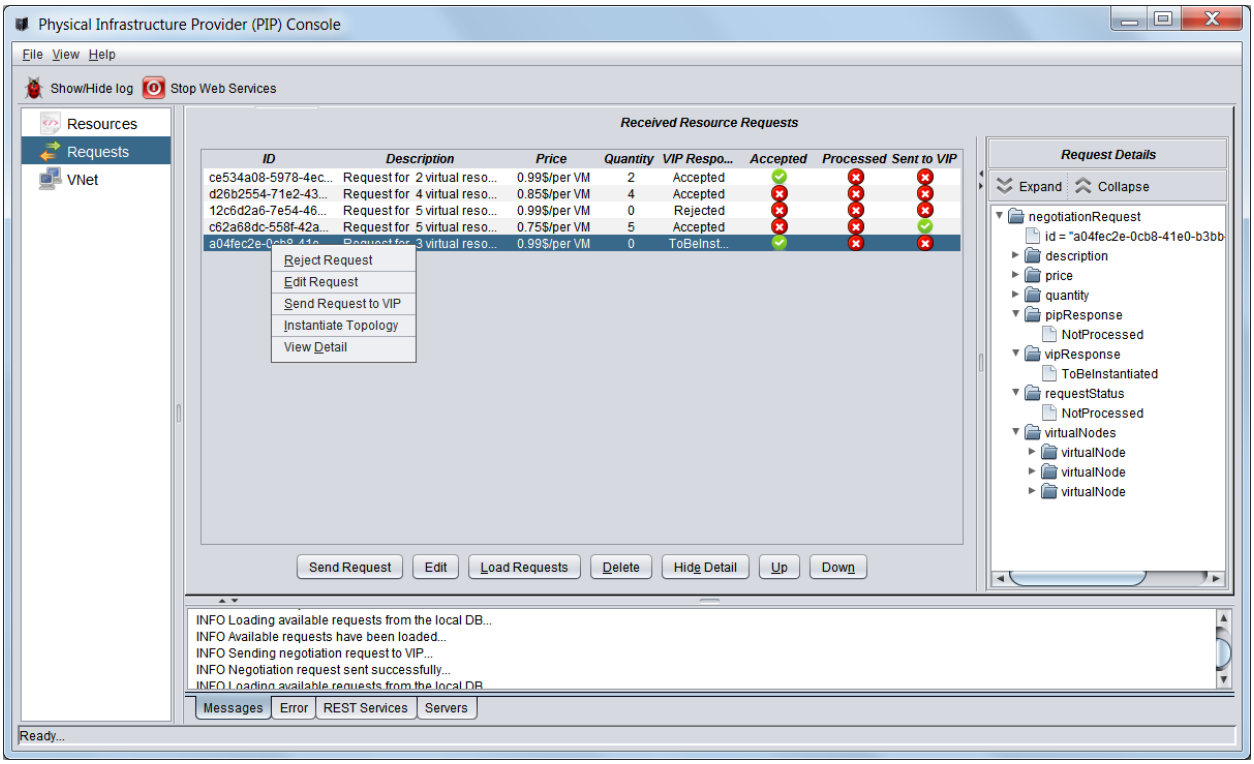

Figure 16: The negotiation interface of the Physical Infrastructure Provider

Figure 17 shows the interfaces intended for the VIP. Whereas Figure 16 shows PIP's resource negotiation interface. Listing D.2 and E.2(Appendix E and Appendix D) show the message logs of the actions taken by the both PIP and VIP components during a negotiation process.

5.4.5 Virtual Topology Instantiation and Resource Management

One of our main goals was to automate the process virtual topology instantiation and the provisioning of resources without human intervention.

The virtual topology instantiation process consists of creating the requested virtual resources and configuring their network settings. After concluding the negotiation process and reaching an agreement, the VIP triggers the instantiation process by changing the negotiation status's request to *ToBeInstantiated*. The request is sent back to the PIP along with the description of the resources and their wanted network settings/configurations such as the number of list virtual interfaces (VIFs) per VM and the IP addresses associated with each interfaces, and

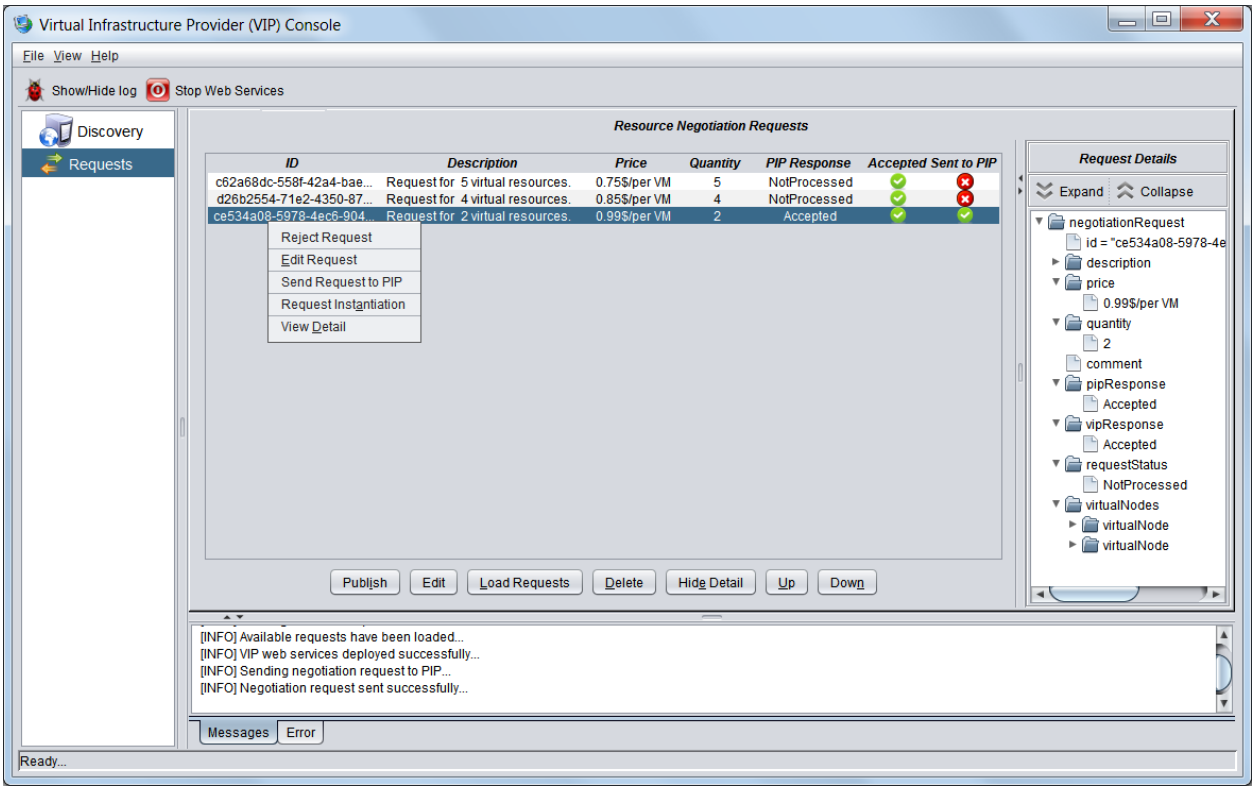

Figure 17: The negotiation interface of the Virtual Infrastructure Provider

so on. The process of creating a VM is similar to the process of creating a physical one that involves assembling and configuring hardware components such as hard disk drive, memory and network cards. To facilitate the VM creation process, we prepared a set of virtual machine templates on which we deployed Shell scripts that enable the remote configuration of the managed VM (addition or removal of Ethernet interface(s), changing a VM's IP address as well as setting/removing a static route between two nodes). These scripts are listed in Appendix B, Listing B.1 and Listing B.2.

Upon receiving the instantiation request, the SM processes the list of resources to create for each of which the SM creates a Java Thread to perform the required XML-RPC calls to create the resource. After successfully creating all the resources, the process of configuring their network settings starts. The SM opens an SSH connection to the targeted virtual machine and to execute the appropriate scripts depending on the type of the configuration required. In addition to creating and configuring virtual resources, the SM monitors

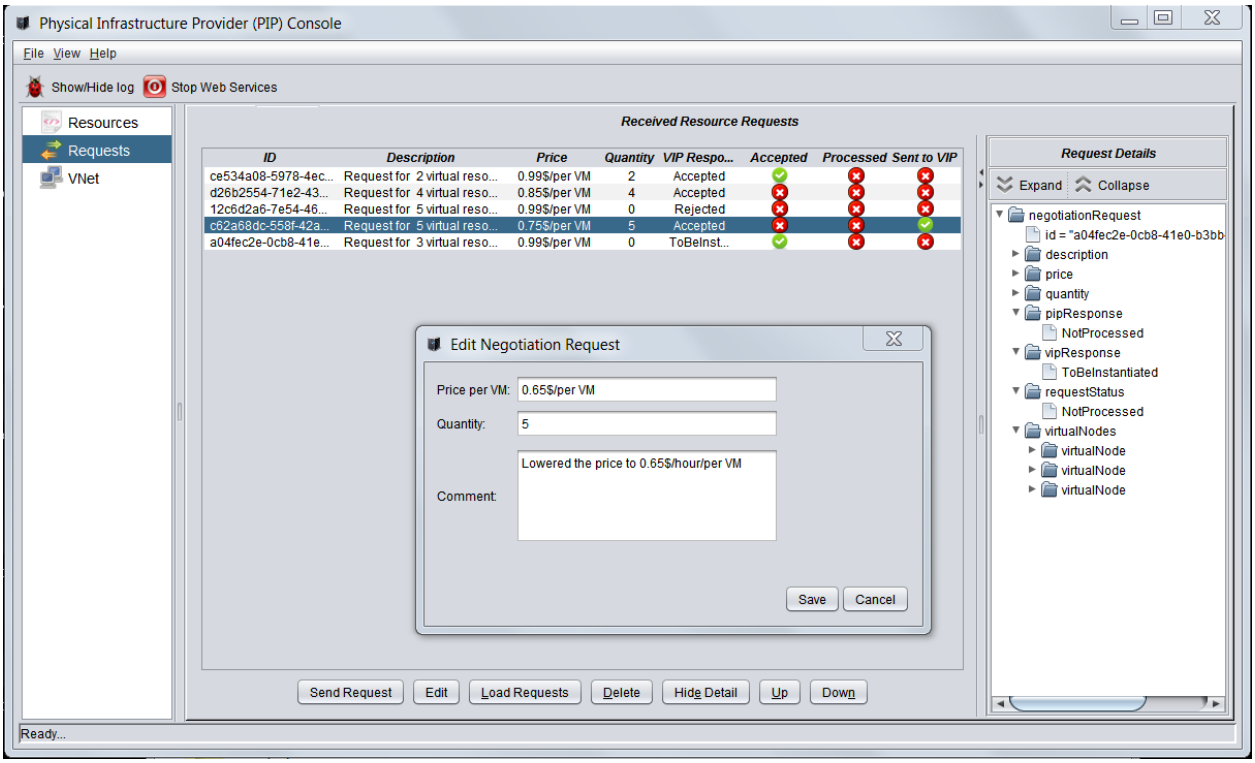

Figure 18: Editing resource negotiation request

the status of the running resources and displays their dynamic attributes on the PIP's interface. Listing D.3 in Appendix D shows the message logs of the actions taken during the instantiation and configuration process. As shown in Figure 19, the left panel of the interface provides a tree view of the managed physical hosts along with their hosted virtual resources. As shown in the figure, a dynamic context menu whose commands depend on the node type (physical, virtual), is associated with each node of the tree. Commands to starts, stop, connect/disconnect a to/from a host as well as view node status for monitoring purposes are provided. The panel located in the center shows the virtual topology. The properties panel (located in the left) shows the status of the connected resources (physical or virtual) Finally, the panel located at the bottom is for logging purposes.

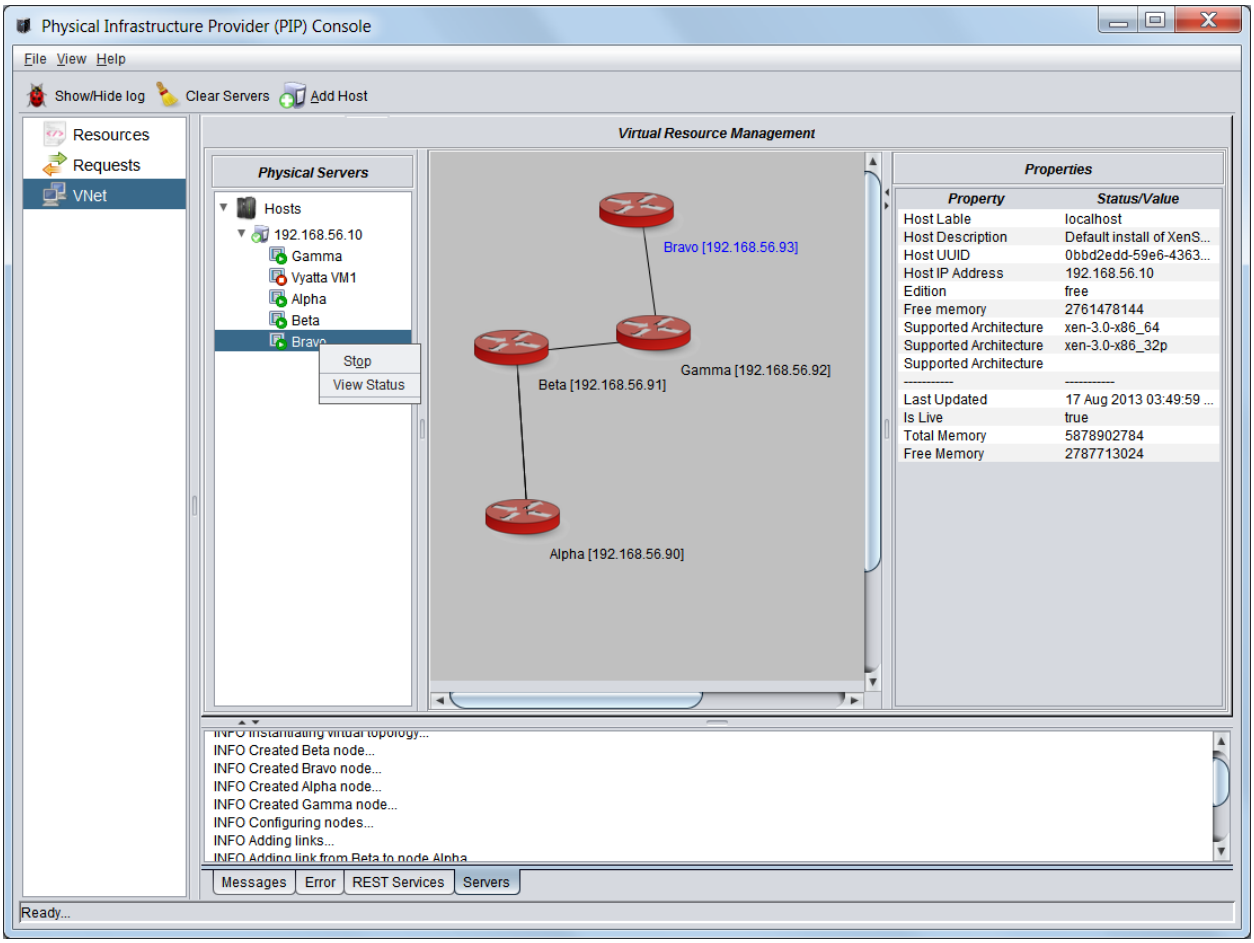

Figure 19: Virtual topology management interface

5.4.6 Broker Components Implementation

5.4.6.1 Resource Selection Algorithm

The implemented selection algorithm is inspirited by the work presented in [89] and listed in 1. The Resource Discovery and Selection Engine (RDS) performs the resource selection and matching process. First, it validates and processes the incoming resource requests, and queries the resource repository and to build a set of candidate resources which later is passed as input to the selection algorithm. Afterwards, it runs the selection algorithm to match each candidate resource with what is the parameters and constraints described in the request.

Listing 5.1 shows an example of a discovery request. It consists of the following key parts:

Selection Parameters – Are used to match a resource's properties. For instance, OS type, node type, virtualization environment, network interface type.

Selection constraint – Has a type and list of parameters. The constraint type specifies the resource's components whose attributes need to be evaluated (e.g CPU, memory, hard drive).

A *constraint parameter* is a function used to assess a particular resource' functional attributes such as CPU's clock speed, memory's size and so on. A key-pair value scheme is used to formulate a particular constraint parameter where the key designates the attribute (e.g number of cores, clock speed) on which the match should be performed. Whereas the value can be one of the following:

- **Fixed value** – Specifies that an exact match should be performed when evaluating the must be equal to the value contained in the parameter.
- **Range** – Of two values (minimum and maximum). This is used to check if the value of the resource's attribute in question is between the specified range.
- **Pattern** – Designates a matching pattern where aggregation function such as Max(*expression*), Sum(*expression*) can be provided. An expression is the formula to be applied on the selected resource's attribute. For example, a Max function can be used to match the highest CPU speed while taking into account the number of cores, the size of its cache memory and its clock speed.

We believe applying selection constraints to the resource selection process as presented in this section results in more flexible and accurate mechanism to select resources.

Algorithm 1 Select best resources that comply with the discovery request

INPUT: A discovery request with the parameters and constraints related to the resources the required

OUTPUT: A list of resources whose properties matched with the constraints specified in the discovery request

```
1: procedure SelectBestResources
2:     candidateSet = null
3:     bestSet = null
4:     candidateSet ← load available resources from DB
5:     for each resource in candidateSet do
6:         evaluate selection parameters on resource
7:         extract the selection constraints from the discovery request
8:         evaluate selection constraints on resource
9:         if resource matches selection parameters and constraints then
10:            add matched resource to BestSet
11:        end if
12:    end for
13:    Return BestSet
14: end procedure
```

Listing 5.1: An example of a resource discovery request

```
<?xml version="1.0" encoding="UTF-8"?>
<discoveryRequest id="c2d1e220-b56a-4d58-87be-090aafc79333">
   <selectionParameters> <vnType>VM</vnType>
      <virEnvironement>XEN</virEnvironement>
      <osType>Linux</osType>
      <interfaceType>Ethernet</interfaceType>
      <networkStack>TCP/IP</networkStack>
      <cpuType>Intel</cpuType>
      <quantity>2</quantity>
      <selectionConstraints>
         <selectionConstraint>
            <constraintOn>Memory</constraintOn>
            <params>
               <param type="size" value="range:min=512|max=1024" />
               <param type="speed" value="value:1333" />
            </params>
         </selectionConstraint>
         <selectionConstraint>
            <constraintOn>CPU</constraintOn>
            <params>
```

```
                <param type="cores" value="range:min=4|max=6" />
                <param type="clockSpeed" value="range:min=1|max=1.7" />
            </params>
        </selectionConstraint>
    </selectionConstraints>
  </selectionParameters>
</discoveryRequest>
```

5.4.6.2 Broker Web Services

Resources being managed	**Base URL** *http://broker.com/api/v1/*	**HTTP Methods description**
virtual and physical resources	/resources	POST: creates a new resource GET: returns a lsit of all resources
	/resources/resource_id	GET: retrieves a resource by its ID PUT: updates a resource's information
	/resources/destroy/resource_id	POST: removes the specified resource
Network services	/services	POST: creates a new service GET: gets a list of all services
	/services/service_id	GET: retrieves a service PUT: updates the specified service
	/services/destroy/service_id	POST: removes the specified service
Role information	/roles	POST: creates a new role GET: gets a lit of all roles information
	/roles/role_id	GET: retrieves a role PUT: updates a role
	/roles/destroy/role_id	POST: deletes an individual role.
virtual network	/networks	POST: creates a new network GET: gets list of all networks
	/networks/network_id	GET: retrieves a network PUT: updates a service
	/networks/destroy/network_id	POST: removes the specified network

Table 3: Broker web services' API

As opposed to Big Web services, RESTful services are not published in a service registry (UDDI) to be later discovered, however, they are available at a uniform paths (under the root URI) that are handed to the requester beforehand, thus, in most cases, provided with the API documentation.

A well-defined URI template should be used to identity entities and illustrate their

relationships. The service and resource (SRR) is an information store that holds information about resources that are arranged in a directory-structure like–from a logical point of view. REST services operate on datasets (nouns), which, in turn, could have sub-dataset(s). Hence, we use the following root URIs to address the respective services: */resources*, */services*, */networks*, */roles*, */requests*. Binding one of these URIs to the base URL (e.g., *http://hostname/api/apiVersion/*) leads to a service's path, e.g the "service" resource is available at *http://broker.com/api/v1.0/services/service_id/*. We notice the base URL has an API version that is used for maintenance proposes. This enables the deployment of new API version and allows the web service clients to bind to a specific version of the API. Although putting the API version in the URI is against REST approach, however, putting it in the resource representation itself is not supported by all the formats (MIME types).
In Table 3, we summarize the uniform interfaces that are used to manage various resources. The resources being managed are listed in the first column, while the second column lists their URI (the uniform interface). We find in the last column the HTTP methods applied on the corresponding URI.

5.4.6.3 Broker User Interface

Figure 20 shows the graphical interface we implemented to interact with the broker subsystem. This interface allows the starting and stopping broker's web services as well as the visualization of information about the published services and resources. The message logs of the broker's components are shown in Listing F.1 and Listing F.2 of Appendix F.

5.5 Use Case–Secure Content Distribution Scenario

Figure 21 illustrates the usage of our proposed information model and architecture for dynamic resource discovery and selection, in a *secure content distribution scenario*. This diagram shows an extended version of the use case we presented in Chapter 3.
The figure depicts the sequence of the interactions between the different components that are involved in the publication, discovery (including selection), and negotiation of resources

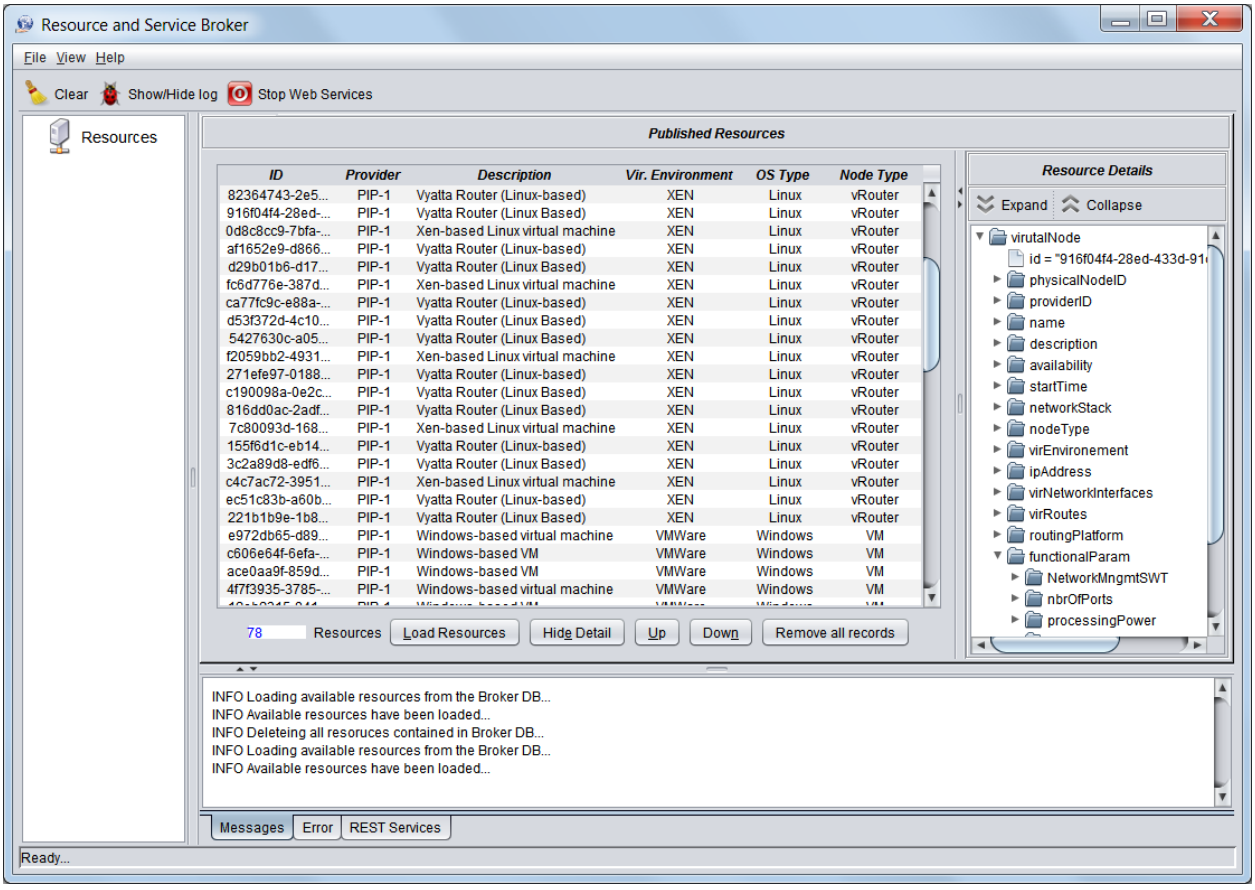

Figure 20: The Broker user interface

as well as the instantiation of virtual network. In addition, this scenario describes the steps that lead to the instantiation of two virtual networks, namely: VNet 1 and VNet2. The components that realize the functions of a given role are grouped together. Although this scenario could be realized using any kind of communication middleware, we have adopted a REST approach in our implementation. Consequently, the interactions between the different entities are REST-based. At first, we assume that the scenario starts when a PIP publishes the information about resources he offers to the broker. Hence, the Resource Publication Engine (RPE) sends a POST request to the broker publication service's URI with the information about resources along with their constraints to create. The publication service creates new resources sends back a confirmation message (200 OK) and the newly created resources' URIs to RPE (steps 1& 2). To deploy service enablers, a VIP needs to

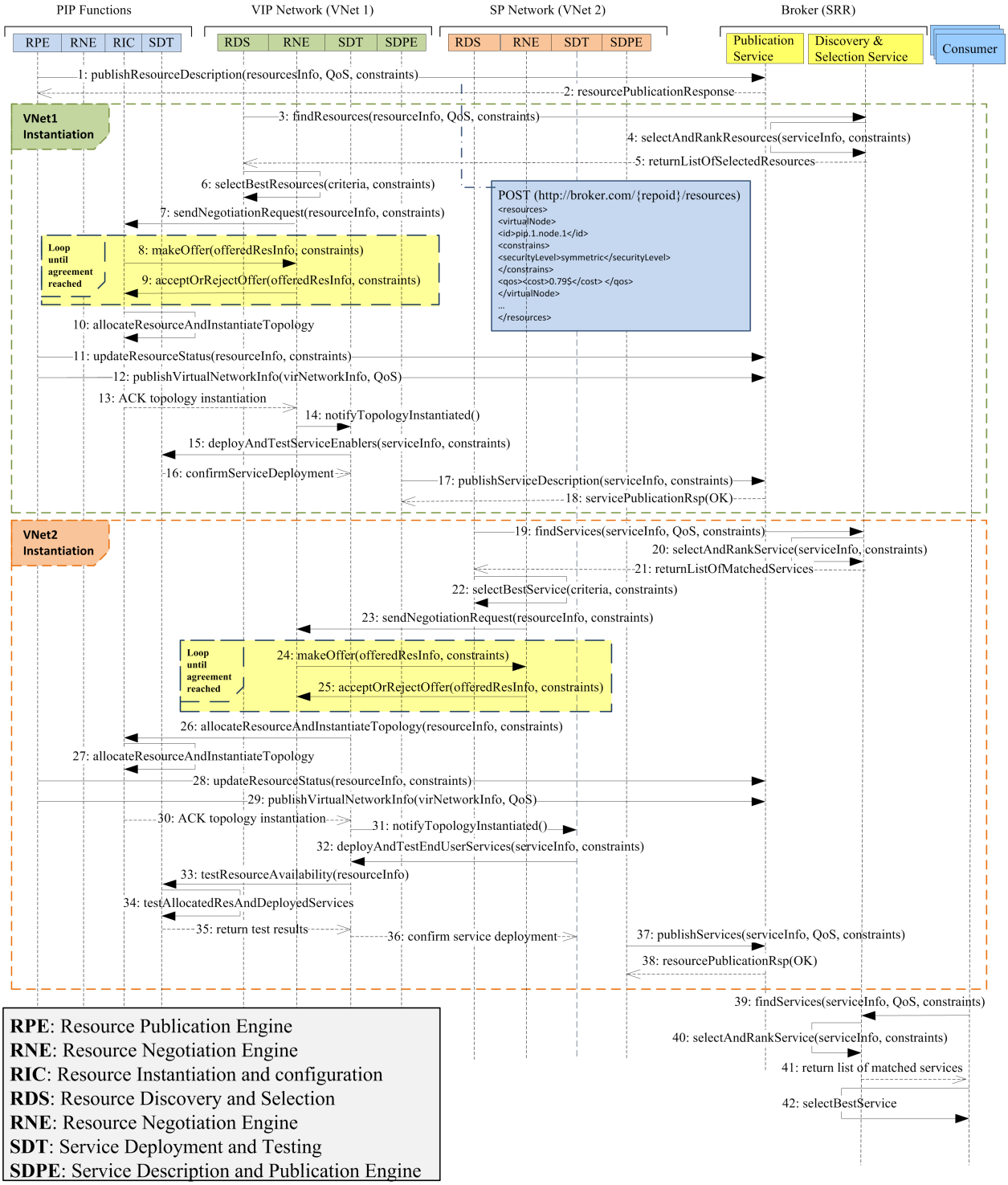

Figure 21: The implemented secure content distribution scenario

create/instantiate a virtual network (i.e VNet1) on top of aggregated resources (possibly from different providers). Therefore, the virtual Resource Discovery and Selection (VRDS) initiates a discovery request containing the description of the desired resources along with their availability and constraints. This request is sent to the broker's Discovery and Selection Service (step 3) which, first, selects the best resources that comply with the requirements specified in the discovery request (using a selection algorithm and with the help of the clustering engine), ranks the selected resources, and replies back with a list of selected resources (steps 4 & 5). In order to refine the received resources, the VRDS performs another selection phase and applies some local criteria and constraints (step 6). Subsequently, the Resource Negotiation Engine (RNE) sends a negotiation request to the corresponding RNE of the lower layer (step 7). The PIP processes the request and sends back an offer with the negotiated resources, which will be later accepted or rejected. Steps 8 and 9 are repeatedly executed until reaching an agreement with the resource requester. The Resource Instantiation and Configuration (RIC) allocates and configures the requested resources, and instantiate the topology while taking into account the specified constraints (step 10). Afterwards, the RPE updates the allocated resources information, and describes as well publishes the newly created virtual network description in the broker. The RIC sends an acknowledgment message confirming the allocated resources to the RNE (at the VIP level), which, in turn, issues a topology instantiated notification that is sent to the Service Deployment and Testing (SDT) (steps 11 to 13). Upon successfully instantiating the virtual topology (resulting in the creation of creating VNet1), the SDT initiates a request for service deployment and test along with the required service information and their constraints, and gets a confirmation message. Finally, SDPE describes the newly created service and publishes its information in the broker (steps 15 to 17). The steps involved in the process of instantiating the topology of VNet2, and the deployment of the content distribution end user service offered by the SP are somehow similar to the steps performed to instantiate VNet1. However, the negotiation process takes place between the SP and the VIP (steps 19 to 38). Furthermore, the content of the message parameters, which determines the type

of the services being offered, and the constraints related to each service are different. Thus, after successfully deploying and testing the end user service, the SDPE sends its description to the broker to be published. Finally, consumers (end-users) who wish to consume end-user services, send a request to the broker for discovering the services of interest. The broker processes the request, selects, and ranks the services that match the initial discovery request (steps 39 to 42). Afterwards, the consumer submits a bind and invoke service request to the chosen SP, which in response sends an acknowledgment and grants access to the consumer. The latter then carries the rest of the interactions related to the end user service invocation and usage (those interactions are not shown in the figure).

5.6 Lessons Learned

We have learned several lessons while implementing the prototype. The first lesson is related to the selection of the database solution for building the resource repositories. Although native XML databases offer better flexibility for storing XML documents, however, they are not suitable for building a large storage repository and their performance related to querying data is questionable in some cases. Therefore, we opted for a hybrid storage solution that combines relational and native XML.

Another lesson related to the choice of virtualization environment and tools. We needed an efficient virtualization solution/tool that provides remote connectivity and exposes programmatic access to the managerial tasks related to the management of virtual resources. Although there are many well-known open source projects such as OpenStack, OpenNebula and CloudStack that provide tools for creating virtual networks, however, such projects require the user interactions to instantiate a virtual topology. Currently, there are two options to have XAPI-enabled virtualization environment. The first one is to install it from an ISO image which results, after installation, in an out-of-the-box virtualization solution with CentOS 5.x based Xen's Dom0. The second one is to install it using XCP-XAPI packages that are currently available on Debian-based Linux distributions, which results in having a Dom0 running Debian-based Kernel. Although installing XCP-XAPI provides more

flexibility in building the virtualization environment, the configuration and setup process is challenging and more complex. Aiming at benefiting from long time support for Ubuntu 12.04, we first installed XCP-XAPI on Ubuntu 12.04.2 LTS, but we discarded this choice due to the lack of automated environment configuration as many workarounds need to be performed in order to configure Xen's environment (such as disabling Xend, downgrading Ubuntu Kernel from 3.5.x to 3.2.x, etc). Besides, some packages were not compatible with latest version of Ubuntu 12.0.4.2 LTS.

In relation to the development of the graphical user interfaces, we learned that it is better to develop a multi-threaded desktop user interfaces that is able to send HTTP requests rather than developing web-based one. Despite its features, web-based interfaces have some limitations, and sometimes the development of such interfaces is harder when it comes to offer an enhanced and better user experience (e.g managing threads and offering a more responsive UI). We use threads for managing each outgoing and incoming request publication and negotiation as well as various resources managerial tasks (such as resource instantiation and monitoring tasks). We learned another lesson related to the selection of a web server for deploying REST web services. We needed a scalable and robust web server that supports Jersey and is able to handle a large number of simultaneous requests. We first selected GlassFish but we discarded this choice after noticing performance issues and moved to Grizzly application server. It uses Java New I/O API and is designed to offer optimum scalability, performance and speed. One of the challenges we have confronted is managing the network connectivity (changing IP address, defining a static route, etc) of Vyatta's routers since we chose the community edition, which is not provided with the REST API that allows to programmatically controlling a router. To prevent this issue from limiting our prototype, we used Shell scripts invoked remotely to execute the necessary commands needed. Although one of the typical solutions to automate the execution of such command a VM is to deploy software agents that are exposed as XML-RPC services, this solution is not flexible in a dynamic networking environment since a VM's IP address might change often.

5.7 Summary

This chapter presented the design and implementation of a proof of concept prototype whose objective is to demonstrate the feasibility of the proposed framework. Although we implemented only a subset the architecture we proposed in Chapter 3, the implemented prototype enables physical infrastructure providers to describe and publish resources into a public resource repository, and virtual infrastructure providers to discover resources and negotiate them with the concerned providers. Additionally, our implementation enables an automated instantiation and configuration of virtual topology. Although many approaches could be used to implement the resource discovery mechanisms (e.g hybrid P2P), we adopted a centralized (client-server) architecture in our implementation. A P2P solution is not predictable since a node can join and leave the network without prior notification, which might be not appropriate to realize the proposed scenario. In this implementation, the resource negotiation process is semi-automated: it requires roles intervention to process a request, and accept or reject it. However, this process can be entirely automated. One solution is to include policies and rules in the negotiation request based on which an automated decision engine can perform the negotiation process. Furthermore, we used SSH and Shell scripts to automate the virtual resources configuration process. Another efficient alternative can be used which consists in deploying software agent that exposes configuring services via public interfaces (XML-RPC or web services).

Chapter 6

Performance and Scalability Evaluation

In the previous chapter, we have presented the design and implementation of a proof of concept prototype that implements a subset of the architecture introduced in Chapter 3 and uses the information model presented in Chapter 4. The implemented prototype enables multiple physical infrastructure providers to describe and publish information about resources into a resource broker as well as virtual infrastructure to discover the published resources and negotiate the selected ones with the respective provider(s).

In this chapter, we focus on evaluation and studying the performance of the implemented system related to resource publication, discovery, negotiation and instantiation. We discuss the performance metrics used, we detail the simulated test scenarios to test the different subsystem and their components, and finally we contrast and discuss the obtained results.

We first assess the basic performance of the individual operations (i.e., publication, discovery, negotiation and virtual topology instantiation in Section 6.1. Then, we evaluate the overall system scalability as presented in Section 6.2.

6.1 Performance Evaluation

We have conducted extensive experiments to evaluate the performance of the components involved in role-to-role interactions in terms of response time, network load and the number of requests handled. We focus on the operations related to resource publication, discovery and selection and negotiation and we measure the response time in milliseconds and network load in bytes. Because of the limited number of physical resources in the testbed we are using, we do not evaluate the maximum size of virtual topology our implementation can handle. Rather, we measure the average time taken to instantiate a virtual machine and elapsed time to configure the virtual network topology. Another fact is due to the limitation associated with the number of virtual machine that a physical machine can host.
The delay measured refers to the total elapsed time from when a request is sent (from one node to another) until a response is sent back to the node that initiated the request. The network load is measured using the Ntopng (NTop Next Generation) [90] and refers to the total number of bytes sent and received between two nodes while processing a request.
By performing these experiments, we do not attempt to prove that we have the best implementation. However, since the response time of the overall system depends on several factors (such as the number of requests to be processed, kind of request, etc), the purpose of the conducted experiments is to demonstrate that the implemented solution is functional.

6.1.1 Prototype Setup

Figure 22 shows the experiment setup used to conduct the basic performance evaluation. In this setup we use three machines (nodes), namely: PIP Management node (PMN), VIP Management node (VMN) and Broker node. Additionally, we configured four nodes to be used as substrate resources. The PMN and VMN are nodes host the implemented components that realize the operations related to PIP and VIP roles respectively. Thus, the Broker node hosts the broker's components and the web services enabling the publication and discovery of resources.

The PMN and VMN and the substrate nodes are DELL Precision 390 machines equally

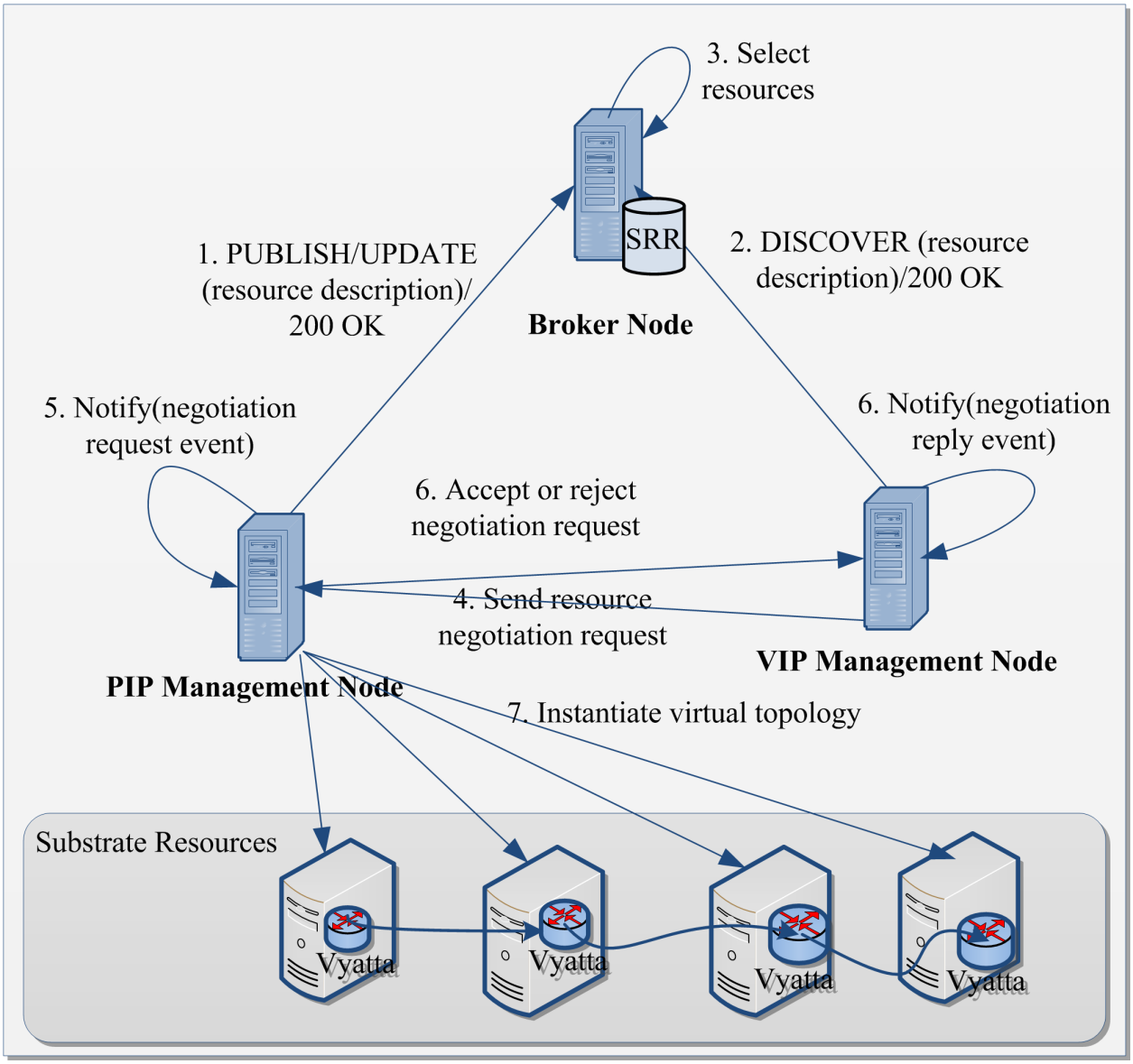

Figure 22: The prototype setup

equipped with Intel Core^{TM}Duo E6550, 2.33GHz processor and 4GB of RAM, 10000 RPM HDD, and 100MBPS link.

Since the Broker node is expected to process all the incoming publication and discovery requests, we used an HP Z210 Workstation machine. It is equipped with Quad Core^{TM}i5 processor, 4GB of RAM (1333 MHz DDR3), 7200 RPM HDD, and 100MBPS link. All the nodes are interconnected with Ethernet links through a Cisco Catalyst 2950 series Switch forming a LAN.

We installed Linux operating system (Ubuntu 12.04 LTS) and the required tools and frameworks on the management and the broker nodes. However, on the remaining four machines, we installed XCP and prepared a set of virtual machines templates configured with 1CPU, 512MB of RAM, 20GB of disk space, and 5Mbps links. In this setup, we run two to four VMs on the same node. Prior to run the experiments, we have generated a set resource description documents containing all the possible resources description to be used during the evaluation process. Such documents were published into the broker using a PUT REST message in order to populate its repository with the required data.

6.1.2 Resource Publication Tests

The objective of this scenario is to test all the components involved in the resource publication process. We test both the PIP and the broker components.

Scenario

In this scenario we use the same setup detailed in Section 6.1.1 to test the case of one single PIP sending publication requests to the broker. Figure 23 shows the two nodes used and the components tested. The *PIP Node* is employed to test the resource publication requests sending process, and *Broker Node* represents the broker and used to test resource publication request handling process.

For us to perform this test, we prepared a script that is used as resource publication simulator and whose message logs are shown in Appendix D, Listing D.4. It uses the Resource

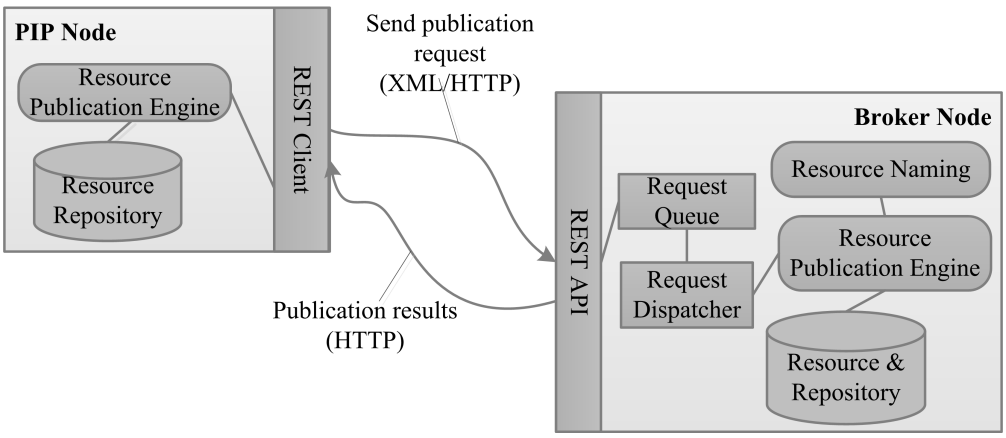

Figure 23: Test scenario for resource publication

Publication Engine to generate any number of publication request and any type of resources. This script continuously sends publication requests with the number of requests specified before execution.

We details the steps and actions performed by each node as follows:

The *PIP Node*: for each request, the *Resource Publication Engine* loads and validates the appropriate resource description document, and asks the REST client to prepare a REST PUT request and send it to the broker.

The *Broker Node*: when received, it extracts the resource description document from the request and processes it. If it is valid, this document is later stored in the database and an HTTP status code (201 created) is sent back to the *PIP Node*.

Results

We measure the overall elapsed time taken to process a resource publication request. The elapsed time is expressed in millisecond and is calculated from the moment a resource document is loaded and sent until a successful response (i.e., HTTP status code 200 OK, or 201 created) is received from the broker. Additionally, the response time includes the time taken to marshal and unmarshal the XML document that contains the resource description to be published, as well as the elapsed time for creating and sending REST request, storing the published resource in the database and sending back the response.

Table 4: Resource publication average network load and response time measurements

Number of publication requests	Response time (ms)	Network Load (kilobyte)
1	245 ms	31.3 KB
50	5006 ms	165.9 KB
100	8845 ms	172.3 KB
150	14773 ms	186.4 KB
300	33003 ms	207.2 KB
400	42319 ms	227.6 KB
500	53005 ms	230.5 KB
600	64478 ms	235.3 KB
1000	109883 ms	239.8 KB

The results shown in Table 4 are the average measurements of 20 tries. The first column shows the number of publication requests per iteration that were repeatedly sent. In this test, we use the same resource description document which consists of the description of one physical machine and two virtual machines. On average, it took 245 ms to process a single publication request which we consider reasonable.

However, as can be concluded, the system performance was affected when the number of publication requests increased, whereas the network load increases slightly. This is due to the request processing overhead and the delay caused by concurrent access to the resource database.

6.1.3 Resource Discovery Tests

The objective of this scenario is to test the resource discovery process and resource selection algorithm.

Scenario

In this scenario, we consider only one single VIP interacting with the broker. Figure 24 shows the broker node and the VIP node that is used to send resource discovery requests. To automate this test, we wrote a script that is used as resource discovery simulator and is

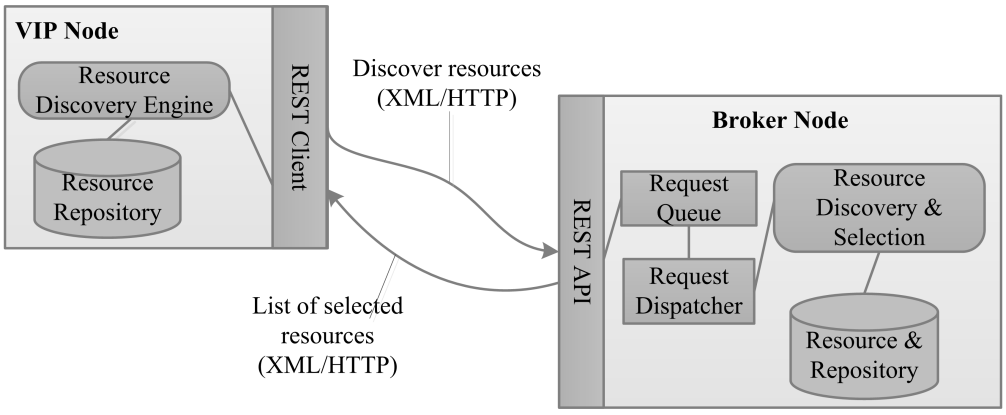

Figure 24: Test scenario for resource discovery

able to send any kind and number of discovery requests. Moreover, this scenario involves the testing of the resource selection algorithm that was previously presented in Chapter 5. The following are the steps and actions performed by each of the aforementioned nodes:

The *VIP Node*: the *Resource Publication Engine* loads and validates the XML documents containing the resource discovery request and sends a message to the REST client module to prepare and send a GET REST request with the loaded resource discovery document.

The *Broker Node*: when received, the broker's *Resource Discovery Engine* extracts the resource discovery document from the request and processes it. If the request is valid, the resource selection algorithm is executed to select the resources that comply with the request. Finally, the list of matched resources is sent to the *VIP Node*.

Results

In this test, we focus on evaluating the performance and efficiency of the resource selection process. We measure the overall elapsed time taken to process a resource discovery request. That is, from the moment a resource discovery request is loaded and sent until receiving a positive response (HTTP 200 OK) with the list of selected resources. Consequently, the response time includes the time taken to marshal and unmarshal the XML document that contains the resource discovery request and sending the GET REST request. Moreover, it includes the overall elapsed time to execute the selection algorithm as well as querying the database to get the list of potential resources and sending back the list of matched resources.

Table 5: Resource discovery network load and response time measurements

Number of discovered resources	Number of resource processed	Response time (ms)	Network Load (kilobyte)
2	50	183 ms	23.2 KB
50	500	1032 ms	294.9 KB
100	500	1885 ms	439.6 KB
200	1000	3548 ms	584.2 KB
400	1300	6958 ms	778.5 KB
800	4000	13744 ms	998.4 KB
1000	5000	17096 ms	14003.9 KB

The message logs of the resource discovery request generator is shown in Appendix E, Listing E.3. To perform the resource discovery experiments, we populated the resource repository with the descriptions of 5000 different resources. The results shown in table 5 are the average response time and network load of 20 tries. The first column shows the number of discovered resources per request. This number is early specified in the discovery request along with the selection parameters and constraints. The second column indicated the number of resources processed during the selection process.

We started by experimenting with a discovery request of two resources. With the setup described earlier in this chapter, it takes 183 ms and generates 23.2 KB network load on average across 20 tries to process the request. As the number of discovered resources increases, the results show that the response time and the network load increase as well. By analyzing the results, we notice that the number of resources processed has an impact on the response which is due to the increased number of resources that are taking into account by the selection algorithm. As for the network load, it increases in size because it depends on the size of the list of matched resources that is sent back to the process who initiated the discovery request.

6.1.4 Resource Negotiation Tests

The purpose of this scenario is to test the operations related to resource negotiation process that involves the PIP and VIP roles. Our main goal is to test the efficiency of the resource negotiation engine with various number or negotiation requests.

Scenario

The nodes used and the components tested are shown in Figure 23. The *VIP Node* is used

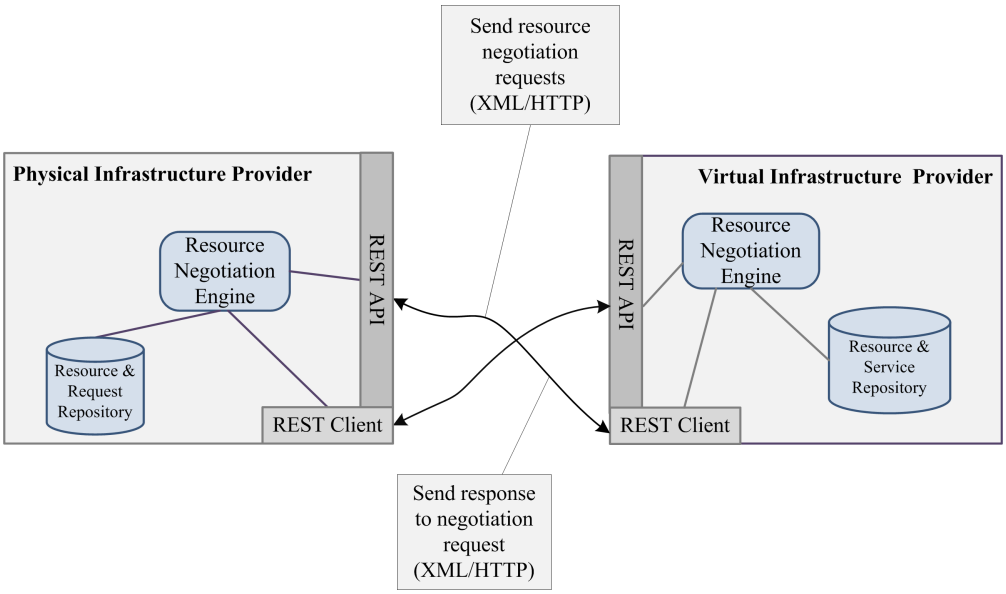

Figure 25: Test scenario for resource negotiation

to generate and send resource negotiation requests, and *PIP Node* is employed to process the request. However, to eliminate the human intervention in processing the request, we modified the Resource Negotiation Engine so that any received request is automatically processed and its status is marked as accepted and returned back to the VIP. The following are the actions performed by each node:

The *VIP Node*: with the help of a script we wrote to generate resource negotiation request, the *Resource Negotiation Engine* loads from and validates the request XML document and sends it to the REST client, which in turns, prepares and sends a REST PUT request to the PIP node.

The *PIP Node*: extracts the negotiation request document that was embedded in the request and processes it. Then, the status of the requests is changed to *processed* and the request is marked as *accepted*. The request is later embedded in a REST PUT request and sent back to the *VIP Node*.

Results

In this scenario, we evaluate the operations involved in the implemented resource negotiation process. Consequently, we measure the overall elapsed time taken to send and process a negotiation request. The response time is expressed in millisecond and is calculated from the moment that the resource negotiation document is retrieved from the resource and request repository and sent to the PIP as well as until a successful response (HTTP status code (201 created) or 200 OK) is received from the PIP.

Table 6: Resource negotiation average network load and response time measurements

Number of negotiation requests	Response time (ms)	Network Load (kilobyte)
1	187 ms	34 .9 KB
50	4711 ms	476.2.9 KB
100	10118 ms	523.6 KB
200	20590 ms	614.9 KB
400	40417 ms	737.4 KB
800	85210 ms	864.2 KB
1000	99466 ms	924.6 KB

Notably, the response time includes the time taken to marshal and unmarshal the XML document consisting in the negotiation request, as well as the elapsed time for creating and sending REST request, storing the published negotiation request as is in the database and sending back the response.

The results shown in table 6 are the average response time and network load of 20 tries. The first column shows the number of negotiation requests generated per iteration. In each request contains information about two virtual resources. On average, it takes 187 ms on to process one single request.

6.1.5 Virtual Topology Instantiation Tests

The objective of this scenario is to test the process of virtual network topology instantiation. In this scenario, we take into consideration the creation of virtual resources (i.e., virtual machines) as well as the instantiation of a virtual network.

Scenario

We use the same setup as detailed in detailed in Section 6.1.1 to execute the

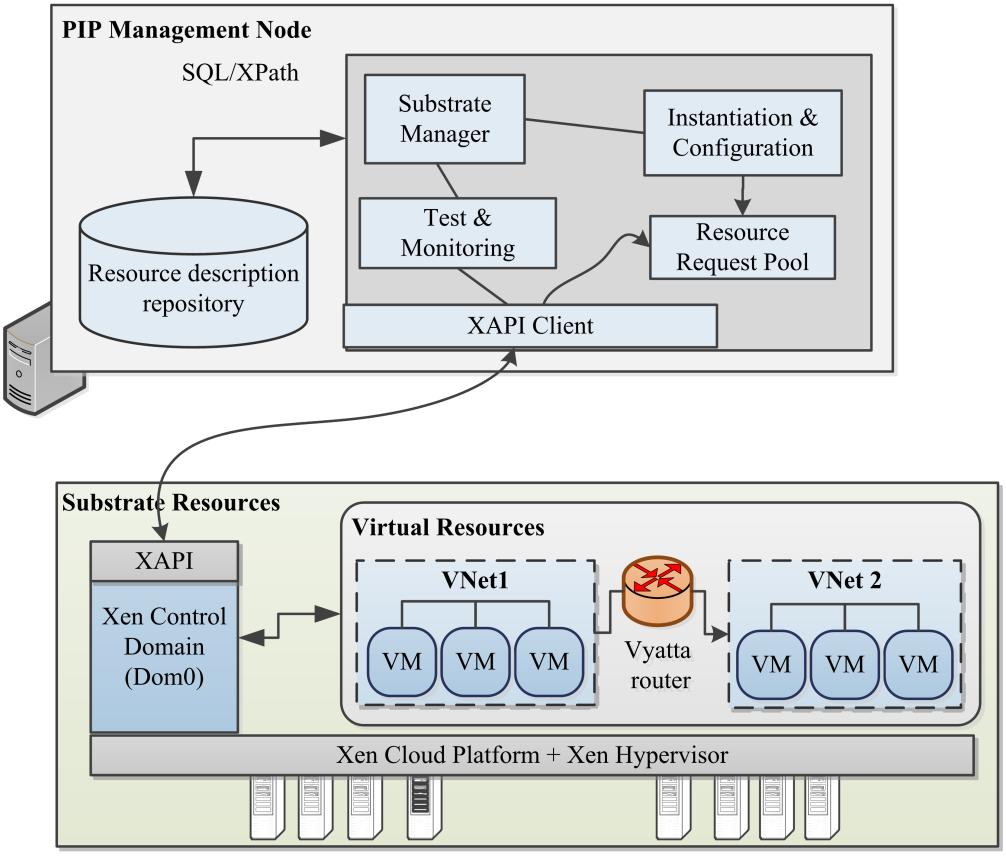

Figure 26: Test scenario for virtual topology instantiation and configuration

experiments. Figure 23 shows the nodes used and the components tested in this scenario. We focus on how the implemented components handle a virtual topology instantiation request. The *PIP Node* is used to communicate with the *Substrate Resources* through the *XAPI Client* module. Whereas the *Substrate Resources* are physical machines we used on which we create and connect virtual machines.

We prepared a script that takes as input a virtual resource instantiation request and uses the *Substrate Manager* to process the request. The request is a document containing the description of resources to instantiation along with their network configuration and setting such as the IP address of each virtual network interface (VIF), the network address, and so on.

The following are the actions and steps tested that are performed by each node: **The *PIP Node*:** the *Substrate Manger* module retrieves a virtual instantiation request that we previously prepared for testing from the request repository, extracts and validates the contained resource descriptions. For each resource, the *Instantiation and Configuration* module Parses resource description and initiates a resource instantiation request that is sent to the *XAPI Client* which in turn communicate with the appropriate physical machine's Dom0 to create the intended virtual resource. Once all virtual resources created, the *Instantiation and Configuration* triggers the resource configuration process which connects the created resources as previously described in the instantiation request.

The *Substrate Resource*: upon receiving virtual machine creation request, the Dom0 of each physical machine configures and creates a virtual machines with the information stated in the request.

Results

We measure the overall time taken to process a virtual topology instantiation request. This time consists of the total time elapsed to create and configure each virtual resource as well as creating and setting up the network interfaces and the routes between two distinct virtual networks.

On average, it takes one minute and 10 seconds to create and configure a Vyatta virtual machine. However, it takes 5 minutes 55 seconds to create and configure a virtual topology consisting of four Vyatta virtual routers.

6.2 Scalability Evaluation of the Implemented System

The goal of the scalability experiments we have conducted is to evaluate how the broker system behaves when is subjected to heavy load. Since the main subsystem is the broker as presented in Chapter 5, the scalability tests focus on the broker's capabilities for handling large number of resource publication and discovery requests.

6.2.1 Scalability Tests Setup

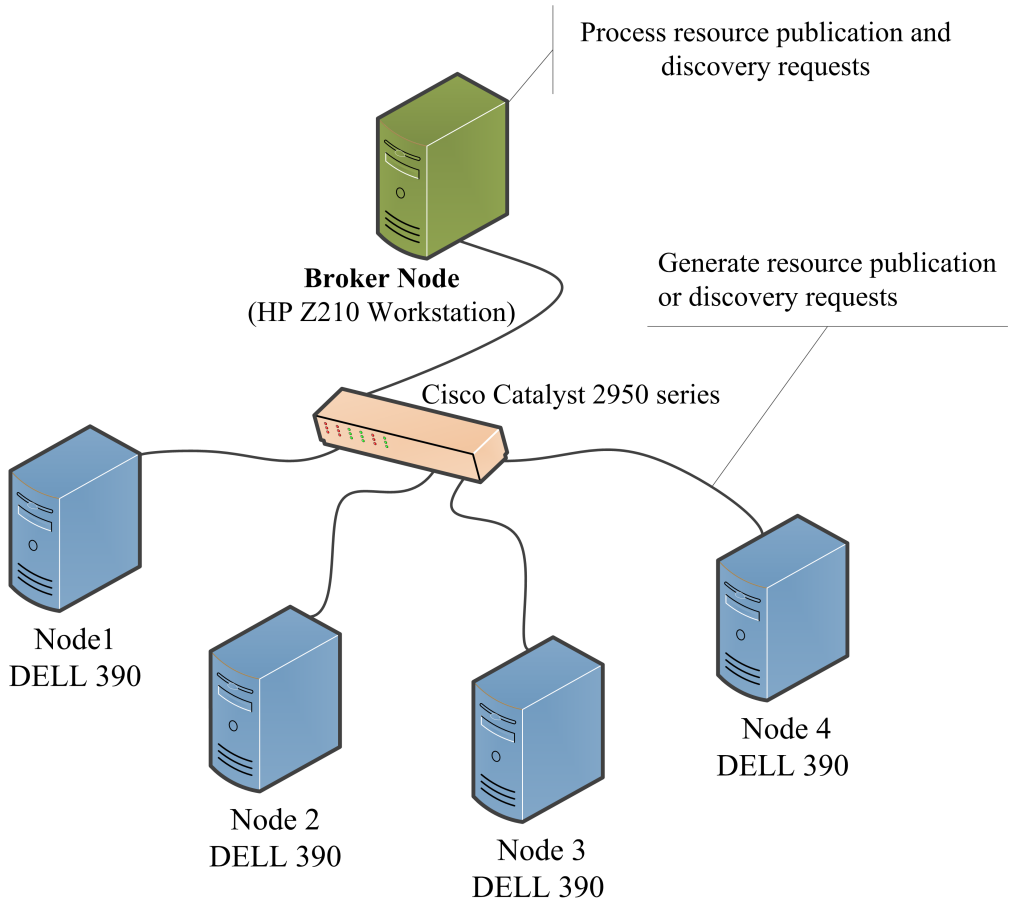

Figure 27: Setup for the scalability tests

The scalability experiments setup is illustrated in Figure 27. The hardware configuration of these machines is detailed in Section 6.1.1. The four DELL 390 machines were used to generate either publication or discovery requests, and the HP Workstation machine was used

as broker node to process the requests.

6.2.2 Resource Publication Scalability Tests

The objective is to assess the capabilities of the broker in handling large number of resource publication requests.

Scenario

In this scenario, we use four PIP nodes and one broker node as depicted in Figure 28 to perform the resource publication experiments. The hardware configuration of these machines is detailed in Section 6.1.1. We focus on studying the ability of the resource broker in handling a large number of concurrent resource publication requests received at the same time. Therefore, to know if the number of resource publication requests to be processed has an impact on the broker's performance, we used the following scenario: we first start the experiments by using only two nodes to generate resource publication requests at the same time; then we scale out to use the four PIP nodes as request generators.

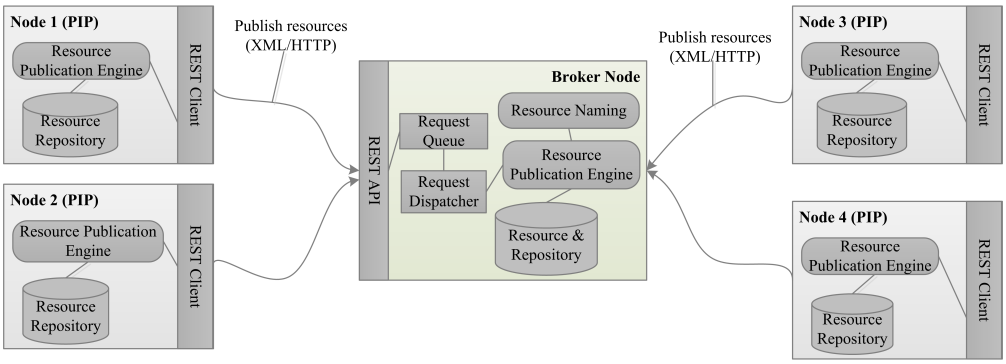

Figure 28: Test scenario for concurrent resource publication

Results

Table 7 shows the average measurements of 20 tries generated with both two nodes and four nodes. The first column illustrates the number of publication requests sent per node, whereas the second and the fifth columns show the total number of published resources per

request. In this scenario, each resource description document used contains a description of two virtual resources. Hence, each publication request consists of two resource descriptions to be processed. For example, the first row shows that 100 publication requests have been sent resulting in 400 as total number or published resources (200 resources per node). We start the experiments by 100 requests and we gradually increases the number of requests. The chart shown in Figure 29 reflects the average response time for resource publication requests. Each line represents test iterations at various loads (number of request). As the number of publication request increases, the response time to process the requests increases in an exponential fashion. This is due to several factors such as database overhead caused by reading/writing records, resource description marshaling and unmarshalling, HTTP requests processing overhead, etc. Consequently, we conclude that average response time to process requests when using four nodes is slightly higher when the number of requests to process is less than 10000. Figure 30 illustrates the average network load for experiments done with both two and four nodes. We conclude that the network load slightly increases as more the number of requests increases as well. Moreover, we notice that the network load increases when more nodes are used to send publication requests. This is because of the additional overhead generated by the packets used to carry resource information.

Table 7: Results of resource publication experiments

	Generated with					
	two nodes			four nodes		
Number of publication requests per node	**Number of published resources**	**Response time (ms)**	**Network Load (kilobyte)**	**Number of published resources**	**Response time (ms)**	**Network Load (kilobyte)**
100	400	18454 ms	197.3 KB	800	32054 ms	250.2 KB
200	800	30402 ms	382.7 KB	1600	41868 ms	426.5 KB
500	2000	77707 ms	412.3 KB	4000	103479 ms	434.3 KB
1000	4000	184442 ms	451.7 KB	8000	188720 ms	537.7 KB
2000	8000	383467 ms	474.3 KB	16000	431953 ms	562.3 KB
3000	12000	576233 ms	520.6 KB	24000	681082 ms	678.2 KB
4000	16000	711442 ms	532 KB	32000	926644 ms	726.8 KB
6000	24000	1192482 ms	565.4KB	48000	1376491 ms	786.3 KB
8000	32000	1508638 ms	626.2 KB	64000	1899391 ms	870.3 KB
10000	40000	1899529 ms	684.6 KB	80000	4702968 ms	948.6 KB
15000	60000	2779286 ms	731.7 KB	120000	7320516 ms	1320.3 KB

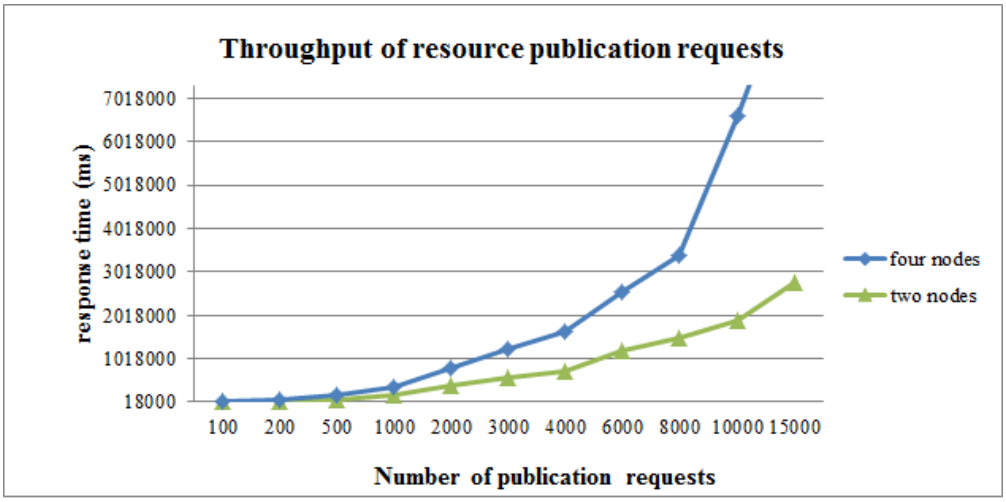

Figure 29: Resource publication response time

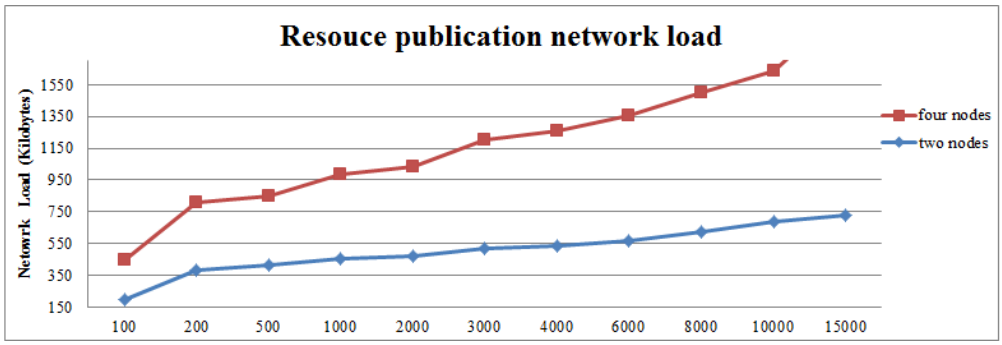

Figure 30: Resource publication network load

6.2.3 Resource Discovery Scalability Test

The objective of these experiments is to test the efficiency of the broker in handling a large number of resource discovery requests. The selection algorithm's assessment is taken into account as well.

Scenario

As depicted in Figure 31, we use four machines (as VIPs) to concurrently send discovery requests to the broker node at the same time. However, we start the experiments by using two VIP nodes. Then, we increase the number of nodes used to four. Thus, we gradually increase the number of resources to be discovered after each try. This gives us insight on the performance of both resource selection algorithm and the implemented logic of resource discovery.

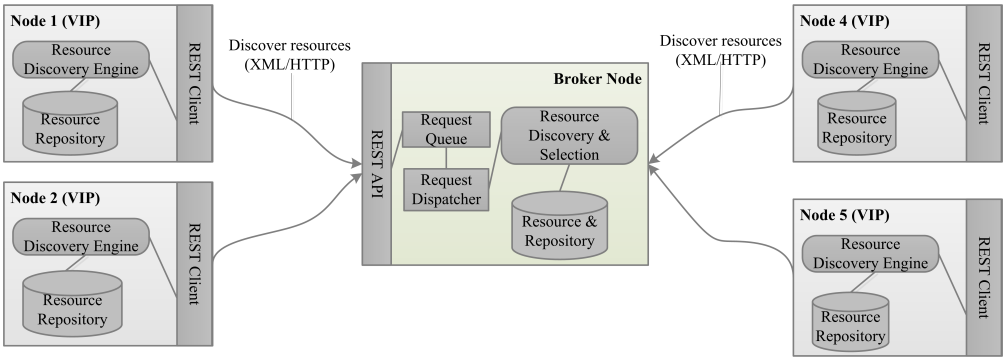

Figure 31: Test scenario for concurrent for resource discovery

Results

The average measurements of 20 tries generated with both two nodes and four nodes are illustrated in Table 8. The first column shows the number of discovered resources per discovery request. As shown in the first row, it takes 329 ms to discover 10 resources using two nodes. However, it takes 1080 ms to discover the same number of resources when we use four nodes. As for the network load, it increases slightly from 112.4 KB to 191.1 KB. Figure 32 reflects the response time of the conducted experiments using both nodes.

Each line represents test operations at various loads. Similarly to resource publication tests, the time taken to process discovery requests increases when the number of resources to discover increases as well. Thus, as shown in Table 8, the implemented broker continued to handle discovery requests of 12000 resources within around 1424096 milliseconds. On other hand, the network load generated by the resource discovery experiments is considerably more important as shown in Figure 33. This is due to the discovered resources' properties that are embedded in the response that is sent back to the VIP node.

Table 8: Scalability results of resource discovery experiments

	Generated with			
	two nodes		**four nodes**	
Number of discovered resources per request	**Response time (ms)**	**Network Load (kilobyte)**	**Response time (ms)**	**Network Load (kilobyte)**
10	329 ms	112.4 KB	1080 ms	190.1 KB
20	692 ms	166.5 KB	2193 ms	294.6 KB
50	2945 ms	358.8 KB	4176 ms	443.2 KB
100	5684 ms	456 KB	7032 ms	667.5 KB
200	11537 ms	766.2 KB	23970 ms	939.8 KB
400	23152 ms	959.9 KB	48371 ms	1234.8 KB
600	34728 ms	1228.8 KB	68630 ms	1433.6 KB
800	46814 ms	1945.6 KB	96665 ms	2340.8 KB
1000	58866 ms	2252.8 KB	114241 ms	2545.4 KB
1500	86944 ms	2560 KB	172194 ms	2983.6 KB
2000	113764 ms	2867.2 KB	236497 ms	3788.8 KB
3000	171210 ms	3584 KB	358506 ms	6656.3 KB
4000	229257 ms	5427.2 KB	477324 ms	9625.6 KB
6000	348910 ms	13619.2 KB	716799 ms	17902.4 KB
8000	449619 ms	14540.8 KB	960802 ms	29081.6 KB
9000	519726 ms	16281.6 KB	1851670 ms	34406.4 KB
12000	694522 ms	20787.2 KB	1424096 ms	45260.8 KB

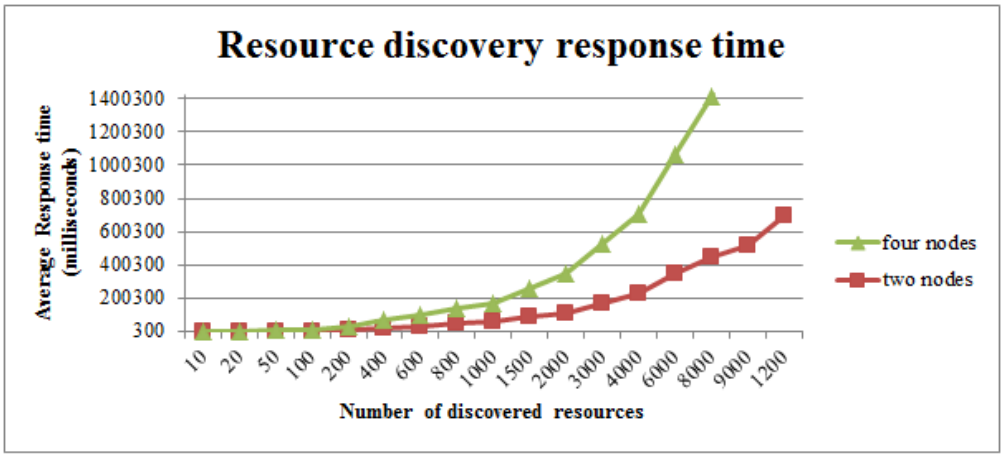

Figure 32: Resource discovery response time

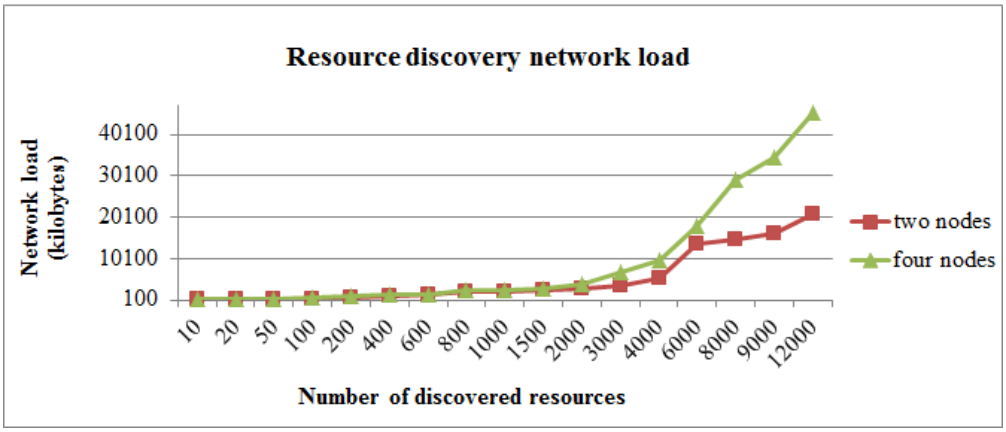

Figure 33: Resource discovery network load

6.3 Summary

This chapter has evaluated the performance and the scalability of the implemented system presented in Chapter 5. We used different test scenarios and test setup to evaluate the resource publication and discovery as well as the resource negotiation and virtual topology instantiation processes. However, we focused on evaluating the performance of the implementation in terms of resource publication and discovery. Therefore, we mainly focused on studying the broker's behavior when it is subjected to normal and large number of requests. By analyzing the result obtained, we conclude that the current implementation gives acceptable results. Although additional processing overhead and increased network load are expected when large number of requests are being processed, many techniques could be used to enhance the overall system performance and reduce the network load. Examples of such techniques include load balancing, data caching, query optimization and the use of a better resource selection algorithm/clustering. Despite the fact that we first used four roles (i.e PIP or VIP) in the experiments, our solution is designed to support multiple roles interacting with each other at the same time. Additionally, our broker implementation has demonstrated to be effective in terms of handling many concurrent requests received simultaneously.

Chapter 7

Conclusions and Future Work

Although network virtualization has received a considerable attention lately and is seen as a promising way to overcome the limitations and fight the gradual ossification of the current Internet infrastructure, it raises many challenges. One of the challenges relates to enabling the dynamic publication, discovery and selection of virtual resources that can be aggregated to form a virtual networks. Another challenge is the definition of an expressive and formal information model that enables the fine-grained description of virtual resources and facilitates the information sharing between the various roles involved. In this chapter, we conclude with a summary of the contributions of this thesis and discuss the planned future work.

7.1 Discussion

We have encountered many challenges in implementing the prototype. Although many selection algorithms have been proposed for network virtualization environment [16, 17, 39], such algorithms have not been implemented and tested in a large-scale environment. One of the challenges we faced is choosing a resource selection algorithm that suits our design requirements. We wanted an efficient selection algorithm that accurately processes a large number of resources. Another challenge we faced is the automation of the virtual topology

instantiation process. The main challenge consisted in creating and configuring virtual resources as described in the instantiation request. Existing solutions require physical infrastructures providers to use command line interface (CLI) or graphical user interface (GUI) to create and configure virtual resources. Thus, using such solutions, one virtual resource is created at a time. However, we wanted a solution that processes the entire set of required virtual resources with minimal human intervention.

7.2 Summary of Contributions

In this thesis, we have presented a framework for resource description, publication and discovery for network virtualization environment.

We first commenced by defining a set of requirements that such framework should meet. Then, we have defined a broker-based architecture that fulfills these requirements. The components of each layer in this architecture are exposed via public interfaces (e.g., web services). A data source repository is used to store information about the published resources. Thus, a selection algorithm has been used to select the best resources that comply with virtual provider's request.

We have proposed a multi-service multi-role integrated information model that enables the fine-grained description of resources and services, and has been published in [9]. First, we have established a set of requirements that the information model should meet. The information model has been extensively used in the implementation to formulate requests such as resource publication, discovery, negotiation and instantiation.

Moreover, we have illustrated the use of the proposed framework in a *secure content distribution scenario*, which illustrates the interactions, and the information exchanged between various roles.

As a proof of concept prototype, we have implemented a subset of the framework while focusing on the description, publication, discovery and negotiation of virtual resources as well as instantiation of virtual topology. Most importantly, all the broker components have been implemented which enabled the publication, discovery and selection of resources provided by various roles.

Finally, we evaluated the performance of the prototype and we have conducted extensive experiments to assess the scalability of the overall system in terms of publication and discovery of resources. The prototype have been tested using different scenarios as detailed in Chapter 6, and the performance measurements taken demonstrated that the use of the proposed framework is acceptable.

7.3 Future Work

The framework we defined provides a solution for dynamic publication and discovery for NVE. However, some improvements and additional implementations can be added to the current prototype implementation.

Although the resource negotiation process requires human intervention, it can be fully automated to be more request-reply oriented based on some predefined policies.

Although the implemented resource selection algorithm has demonstrated to be effective during the performance evaluation and scalability assessment, it can be enhanced to support resource clustering and matching.

The proposed framework was evaluated using a limited set of resources. As future work, we plan to evaluate the framework in a larger environment involving a considerable amount of computational resources. One option to perform such evaluation is to use Amazon Elastic Compute Cloud (Amazon EC2) [26] or use Planetlab cloud services or PlantLab's resources

[32]. The use of such infrastructure allows for simulating the results a in a real-world-like environment.

As per design, the proposed architecture is broker based, hence centralized. Another alternative is to decentralize the broker services using a peer-to-peer approach.

References

[1] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, 2009.

[2] J. Turner and D. Taylor, "Diversifying the internet," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 2, pp. 6 pp.–760, 2005.

[3] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 61–64, Jan. 2007.

[4] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. Correia, "The way 4ward to the creation of a future internet," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pp. 1–5, 2008.

[5] L. Correia, H. Abramowicz, M. Johnsson, and K. Wünstel, *Architecture and Design for the Future Internet: 4WARD Project*. Signals and Communication Technology, Springer, 2011.

[6] M. El Barachi, N. Kara, and R. Dssouli, "Open virtual playground: Initial architecture and results," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pp. 576 –581, jan. 2012.

[7] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 22–31, Oct. 2009.

[8] N. Niebert, I. Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs, "Network virtualization: A viable path towards the future internet," *Wireless Personal Communications*, vol. 45, no. 4, pp. 511–520, 2008.

[9] M. El Barachi, S. Rabah, N. Kara, R. Dssouli, and J. Paquet, "A multi-service multi-role integrated information model for dynamic resource discovery in virtual networks," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pp. 4777–4782, 2013.

[10] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, pp. 277–288, Nov. 1984.

[11] M. Böhm, S. Leimeister, C. Riedl, and H. Krcmar, "Cloud computing - outsourcing 2.0 or a new business model for it provisioning?," in *Application Management* (F. Keuper, C. Oecking, and A. Degenhardt, eds.), pp. 31–56, Gabler, 2011.

[12] J. Carapinha and J. Jiménez, "Network virtualization: a view from the bottom," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, VISA '09, (New York, NY, USA), pp. 73–80, ACM, 2009.

[13] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[14] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network virtualization architecture: proposal and initial prototype," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, VISA '09, (New York, NY, USA), pp. 63–72, ACM, 2009.

[15] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, "Network virtualization: a hypervisor for the internet?," *Communications Magazine, IEEE*, vol. 50, no. 1, pp. 136–143, 2012.

[16] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke, "Virtual resource description and clustering for virtual network discovery," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pp. 1–6, 2009.

[17] H. Medhioub, I. Houidi, W. Louati, and D. Zeghlache, "Design, implementation and evaluation of virtual resource description and clustering framework," in *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, pp. 83–89, 2011.

[18] P. H. Gum, "System/370 extended architecture: Facilities for virtual machines," *IBM Journal of Research and Development*, vol. 27, no. 6, pp. 530–544, 1983.

[19] Microsoft, "Microsoft hyper-v." [online] http://www.microsoft.com/en-us/server-cloud/hyper-v-server/default.aspx. Accessed March, 2013.

[20] VMWare, "Vmware vsphere." [online] http://www.vmware.com/products/vsphere/. Accessed March, 2013.

[21] Citrix, "Citrix xenserver." [online] http://www.citrix.com/products/xenserver/overview.html/. Accessed March, 2013.

[22] K. A. Scarfone, M. P. Souppaya, and P. Hoffman, "Sp 800-125. guide to security for full virtualization technologies," tech. rep., Gaithersburg, MD, United States, 2011.

[23] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 164–177, Oct. 2003.

[24] C. Edwards and A. Harwood, "Using para-virtualization as the basis for a federated planetlab architecture," in *Virtualization Technology in Distributed Computing, 2006. VTDC 2006. First International Workshop on*, pp. 13–13, 2006.

[25] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[26] Amazon, "Amazon elastic compute cloud." [online] http://aws.amazon.com/ec2/. Accessed October, 2012.

[27] Google, "Google compute engine." [online] https://cloud.google.com/products/compute-engine. Accessed October, 2012.

[28] IBM, "Ibm infrastructure as a service." [online] http://www-935.ibm.com/services/ca/en/cloud-enterprise/. Accessed October, 2012.

[29] Microsoft, "Windows azure." [online] http://www.windowsazure.com/en-us/. Accessed October, 2012.

[30] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 1114–1128, 2012.

[31] J. Ding, I. Balasingham, and P. Bouvry, "Management of overlay networks: A survey," in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009. UBICOMM '09. Third International Conference on*, pp. 249–255, 2009.

[32] PlanetLab, "Planetlab project." [online] https://www.planet-lab.org/. Accessed October, 2012.

[33] J. Touch, "Dynamic internet overlay deployment and management using the x-bone," in *Computer Networks*, pp. 117–135, 2001.

[34] VINI Project, "A virtual network infrastructure (vini)." [online] http://www.vini-veritas.net/. Accessed October, 2012.

[35] N. Chowdhury, F. Zaheer, and R. Boutaba, "imark: An identity management framework for network virtualization environment," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, pp. 335–342, 2009.

[36] Nouveau Project, "Network virtualization project (nouveau)." [online] http://netlab.cs.uwaterloo.ca/virtual/. Accessed October, 2012.

[37] TINA, "Tina business model and reference points, v.4.0." [online] http://www.tinac.com/specifications/documents/bm_rp.pdf. Accessed June, 2012.

[38] M. El Barachi, N. Kara, and R. Dssouli, "Towards a service-oriented network virtualization architecture," in *Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services, 2010 ITU-T*, pp. 1–7, 2010.

[39] H. Amarasinghe, A. Belbekkouche, and A. Karmouch, "Aggregation-based discovery for virtual network environments," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 1276–1280, 2012.

[40] F.-E. Zaheer, J. Xiao, and R. Boutaba, "Multi-provider service negotiation and contracting in network virtualization.," in *NOMS*, pp. 471–478, IEEE, 2010.

[41] P. Rygielski and S. Kounev, "Network virtualization for qos-aware resource management in cloud data centers: A survey," *Praxis der Informationsverarbeitung und Kommunikation*, vol. 36, no. 1, pp. 55–64, 2013.

[42] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, 2013.

[43] A. Beben, P. Wisniewski, P. Krawiec, M. Nowak, P. Pecka, J. Batalla, P. Bialon, P. Olender, J. Gutkowski, B. Belter, and L. Lopatowski, "Content aware network based on virtual infrastructure," in *Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing (SNPD), 2012 13th ACIS International Conference on*, pp. 643–648, 2012.

[44] S. Balasubramaniam, J. Mineraud, P. Perry, B. Jennings, L. Murphy, W. Donnelly, and D. Botvich, "Coordinating allocation of resources for multiple virtual iptv providers to maximize revenue," *Broadcasting, IEEE Transactions on*, vol. 57, no. 4, pp. 826–839, 2011.

[45] F. Belqasmi, R. Glitho, and C. Fu, "Restful web services for service provisioning in next-generation networks: a survey," *Communications Magazine, IEEE*, vol. 49, no. 12, pp. 66–73, 2011.

[46] A. Alamri, M. Eid, and A. E. Saddik, "Classification of the state-of-the-art dynamic web services composition techniques," *Int. J. Web Grid Serv.*, vol. 2, pp. 148–166, Sept. 2006.

[47] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, WWW '08, (New York, NY, USA), pp. 805–814, ACM, 2008.

[48] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, CA, USA, 2000.

[49] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things.," in *IOT* (F. Michahelles and J. Mitsugi, eds.), IEEE, 2010.

[50] B. Murugan and D. Lopez, "Article: A survey of resource discovery approaches in distributed computing environment," *International Journal of Computer Applications*, vol. 22, pp. 44–46, May 2011. Published by Foundation of Computer Science.

[51] X. Wang and L. fu Kong, "Resource clustering based decentralized resource discovery scheme in computing grid," in *Machine Learning and Cybernetics, 2007 International Conference on*, vol. 7, pp. 3859–3863, 2007.

[52] B. Lv, Z. Wang, T. Huang, J. Chen, and Y. Liu, "Virtual resource organization and virtual network embedding across multiple domains," in *Proceedings of the 2010 International Conference on Multimedia Information Networking and Security*, MINES '10, (Washington, DC, USA), pp. 725–728, IEEE Computer Society, 2010.

[53] Y. Xu, Y. Han, W. Niu, Y. Li, T. Lin, and S. Ci, "A reference model for virtual resource description and discovery in virtual networks," in *ICCSA (3)*, pp. 297–310, 2012.

[54] M. Serhani, R. Dssouli, A. Hafid, and H. Sahraoui, "A qos broker based architecture for efficient web services selection," in *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pp. 113–120 vol.1, 2005.

[55] I. Fajjari, M. Ayari, and G. Pujolle, "Vn-sla: A virtual network specification schema for virtual network provisioning," in *Networks (ICN), 2010 Ninth International Conference on*, pp. 337–342, 2010.

[56] J. Osullivan, *Towards a Precise Understanding of Service Properties*. PhD thesis, Queensland University of Technology, 2006.

[57] G. Koslovski, P.-B. Primet, and A. CharÃ£o, "Vxdl: Virtual resources and interconnection networks description language," in *Networks for Grid Applications* (P. Vicat-Blanc Primet, T. Kudoh, and J. Mambretti, eds.), vol. 2 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 138–154, Springer Berlin Heidelberg, 2009.

[58] The VGrADS Project, "Virtual grid description language (vgdl)." [online] http://vgrads.rice.edu/research/execution_system/virtual_grids/vgdl/. Accessed June, 2012.

[59] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Distributed resource discovery on PlanetLab with SWORD," 2004.

[60] GLUE Working Group, "Glue working group (glue-wg)." [online] http://www.ogf.org/documents/GFD.147.pdf. Accessed June, 2012.

[61] D. Zhou, L. Zhong, T. Wo, and J. Kang, "Cloudview: Describe and maintain resource view in cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 151–158, 2010.

[62] OASIS, "Data center markup language (dcml)." [online] http://www.dcml.org/. Accessed October, 2012.

[63] D. Zhou, L. Zhong, T. Wo, and J. Kang, "Cloudview: Describe and maintain resource view in cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 151–158, 2010.

[64] J. Van Der Ham, F. Dijkstra, P. Grosso, R. Van Der Pol, A. Toonk, and C. De Laat, "A distributed topology information system for optical networks based on the semantic web," *Opt. Switch. Netw.*, vol. 5, pp. 85–93, June 2008.

[65] A. Campi and F. Callegati, "Network resource description language," in *GLOBECOM Workshops, 2009 IEEE*, pp. 1–6, 2009.

[66] Open Grid Forum, "Network mark-up language working group (nml-wg)." [online] http://www.ogf.org/gf/group_info/view.php?group=nml-wg. Accessed Jun, 2012.

[67] H. Wang, "Nevml: A markup language for network topology visualization," in *Future Networks, 2010. ICFN '10. Second International Conference on*, pp. 119–123, 2010.

[68] DMTF, "Common information model." [online] http://dmtf.org/standards/cim. Accessed Jun, 2012.

[69] GÉANT2, "Common network information service schema specification." [online] http://geant2.net. Accessed August, 2012.

[70] J. Strassner, "Den-ng: achieving business-driven network management," in *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pp. 753–766, 2002.

[71] D. Dobrilovic, Z. Stojanov, B. Odadzic, and B. Markoski, "Using network node description language for modeling networking scenarios," *Adv. Eng. Softw.*, vol. 43, pp. 53–64, Jan. 2012.

[72] M. Ghijsen, J. van der Ham, P. Grosso, and C. de Laat, "Towards an infrastructure description language for modeling computing infrastructures," in *Proceedings of the 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with*

Applications, ISPA '12, (Washington, DC, USA), pp. 207–214, IEEE Computer Society, 2012.

[73] GENI Project, "Rspec (geni)." [online] http://groups.geni.net/geni/wiki/GeniRspec. Accessed September, 2012.

[74] Network Working Group, "Ietf vnmim." [online] http://tools.ietf.org/html/draft-okita-ops-vnetmodel. Accessed October, 2012.

[75] G. Schaffrath, S. Schmid, I. Vaishnavi, A. Khan, and A. Feldmann, "A resource description language with vagueness support for multi-provider cloud networks," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pp. 1–7, 2012.

[76] L. Lymberopoulos, P. Grosso, C. Papagianni, D. Kalogeras, G. Androulidakis, J. Van der Ham, C. De Laat, and V. Maglaris, "Managing federations of virtualized infrastructures: A semantic-aware policy based approach," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp. 1235–1242, 2011.

[77] A. Charfi, B. Schmeling, F. Novelli, H. Witteborg, and U. Kylau, "An overview of the unified service description language," in *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pp. 173–180, 2010.

[78] World Wide Web Consortium (W3C), "Web application description language." [online] http://www.w3.org/Submission/wadl/. Accessed September, 2012.

[79] Martin D, Burstein M, Hobbs J, et al., "Owl-s: Semanic markup for web services." [online] http://www.w3.org/Submission/OWL-S/. Accessed October, 2012.

[80] RDF Working Group, "Resource description framework (rdf)." [online] http://www.w3.org/RDF/. Accessed October, 2012.

[81] I. Baldine, Y. Xin, D. Evans, C. Heerman, J. Chase, V. Marupadi, and A. Yumerefendi, "The missing link: Putting the network in networked cloud computing," in *in ICVCI09: International Conference on the Virtual Computing Initiative*, 2009.

[82] Y. T. Lee, "Information modeling: From design to implementation," in *Proceedings of the Second World Manufacturing Congress*, pp. 315–321, 1999.

[83] Jersey, Project, "Restful web services in java." [online] http://jersey.java.net/. Accessed January, 2013.

[84] Grizzly Project, "Grizzly web server." [online] https://grizzly.java.net/. Accessed January, 2013.

[85] JAXB Project, "Jaxb project." [online] https://jaxb.java.net/. Accessed January, 2013.

[86] PostgreSQL Project, "Postgresql database." [online] http://www.postgresql.org/. Accessed February, 2013.

[87] Xen Project, "Xen cloud platform." [online] http://www.xenproject.org/. Accessed March, 2013.

[88] KVM Project, "Kernel based virtual machine." [online] http://www.linux-kvm.org/page/Main_Page/. Accessed March, 2013.

[89] C. Liu, L. Yang, I. Foster, and D. Angulo, "Design and evaluation of a resource selection framework for grid applications," in *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pp. 63–72, 2002.

[90] NTop Project, "Ntopng." [online] http://www.ntop.org/. Accessed October, 2013.

Part I

Appendices

Appendices

Appendix A

Enumeration Types

A.1 Enumerations Types for Network Nodes

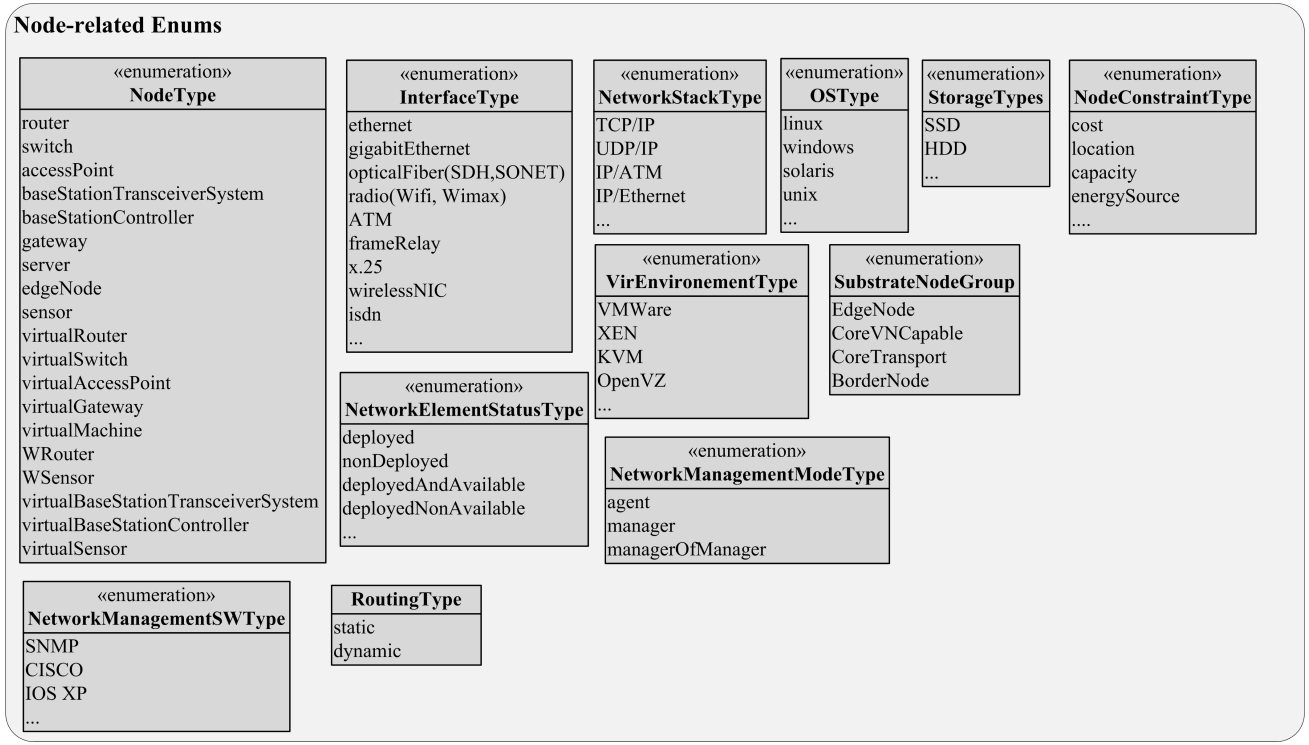

Figure 34: Enumeration types for physical and virtual nodes

A.2 Enumerations Types for Network Links

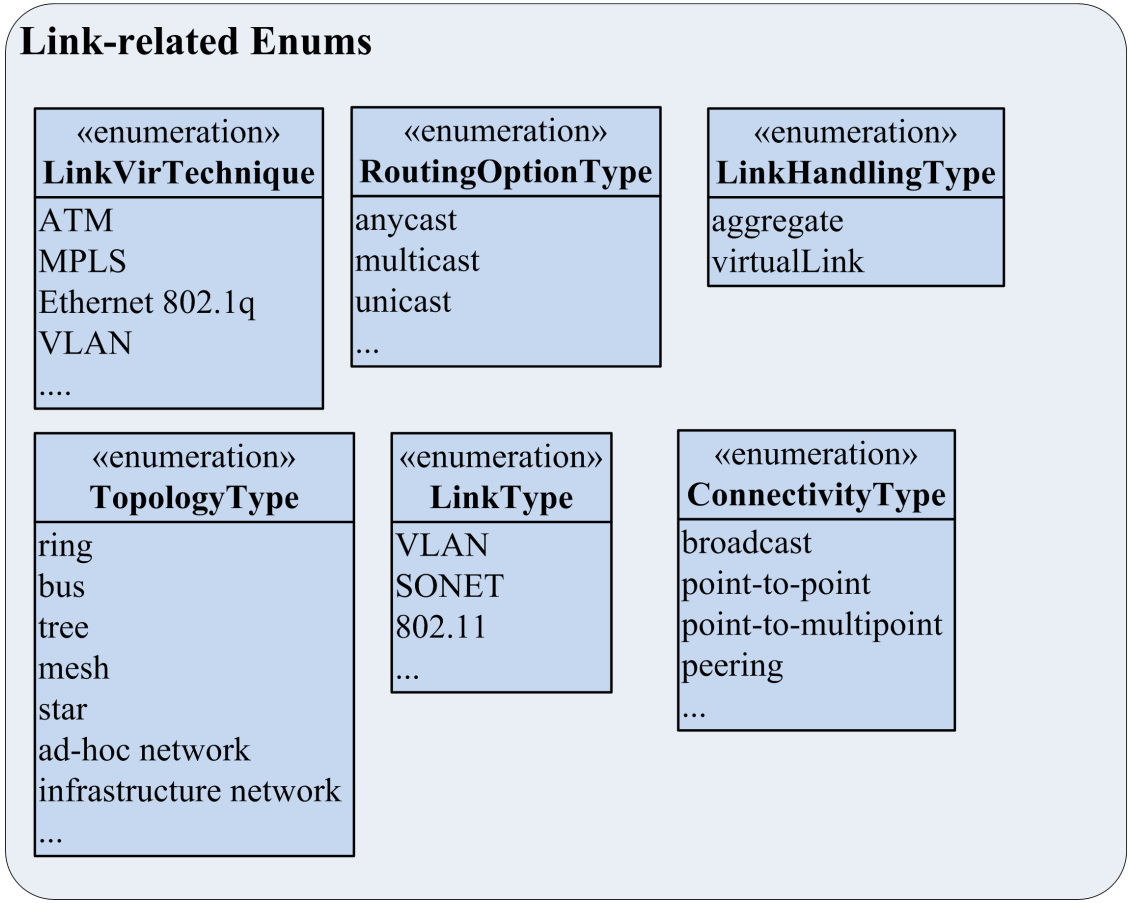

Figure 35: Enumeration types for network physical and virtual links

A.3 Enumerations Types for Network Services

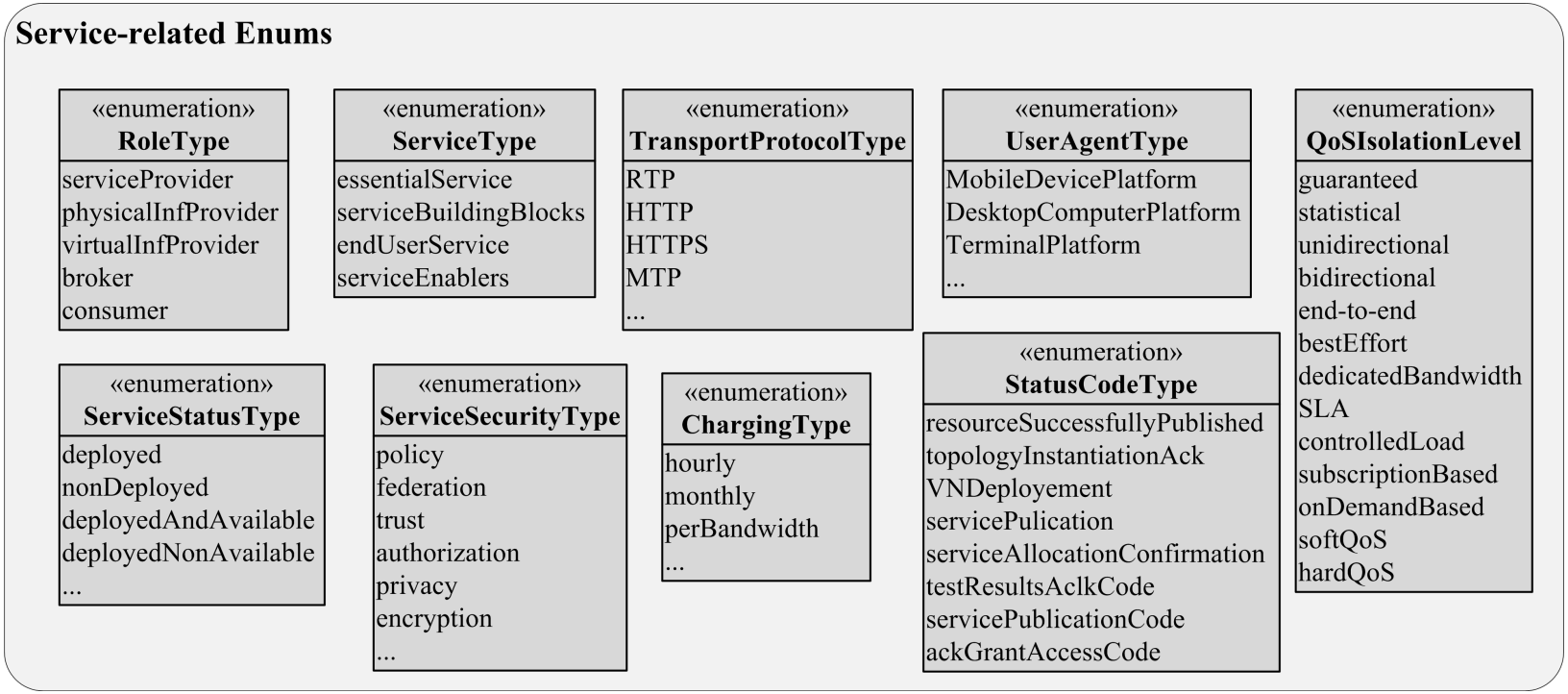

Figure 36: Enumeration types for network-related services

A.4 Security-related Enumerations Types

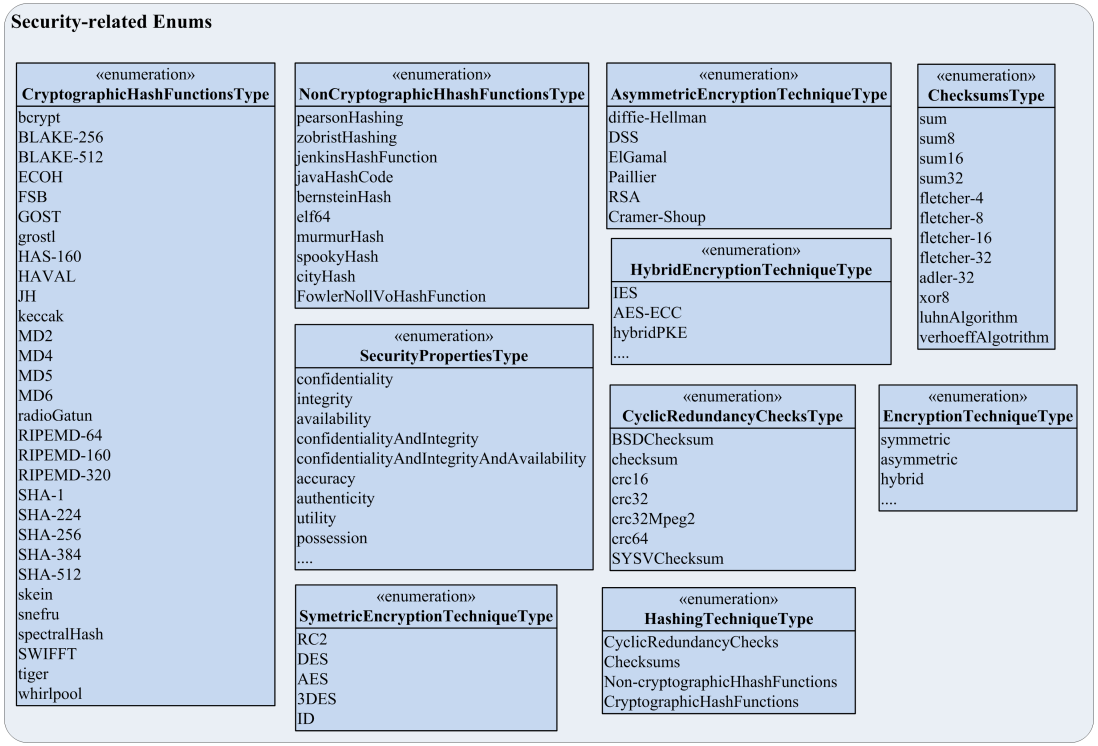

Figure 37: Enumeration types for formulating security-related attributes

A.5 Enumerations for Wireless-related Entities

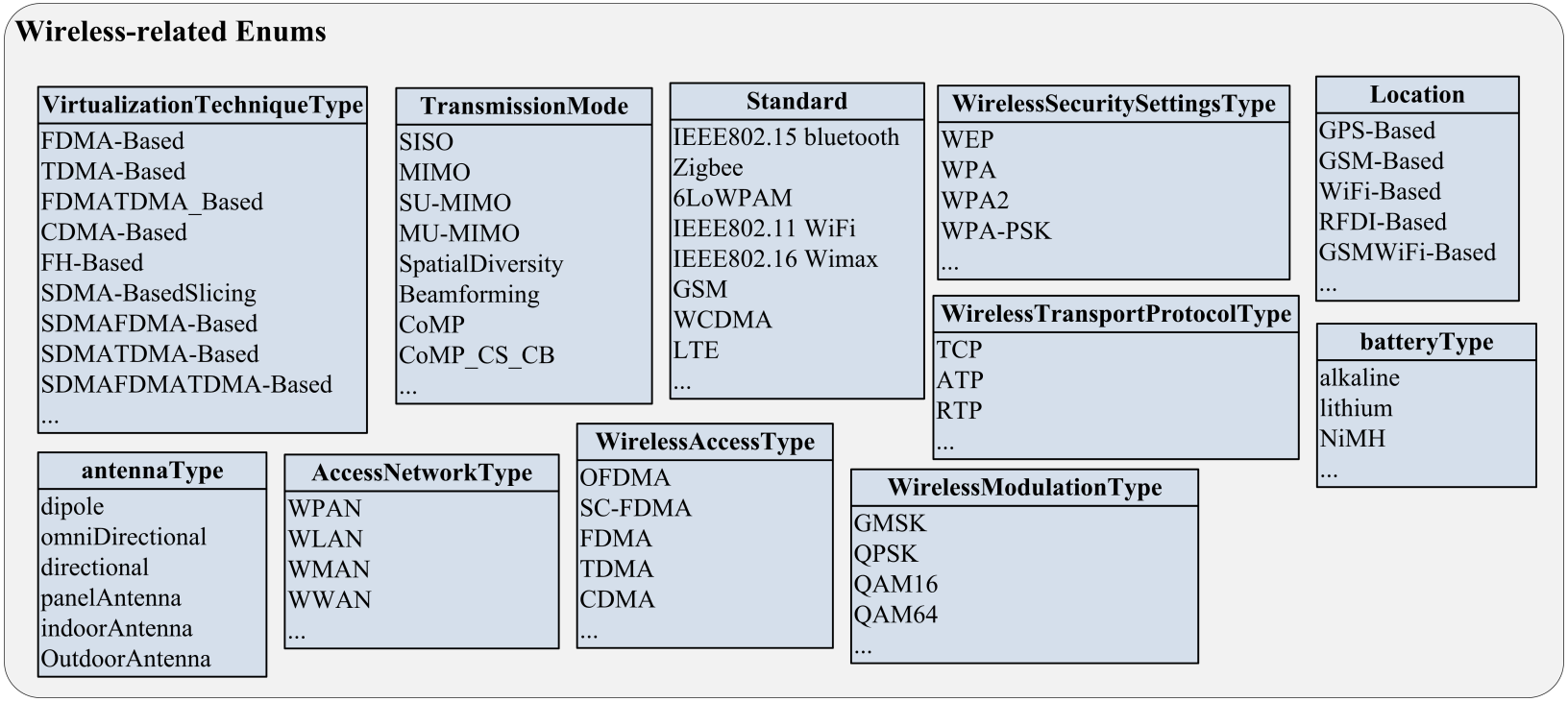

Figure 38: Wireless-related enumeration types

Appendix B

Shell Scripts

B.1 Script for Managing Ethernet Vyatta Virtual Network Interfaces

Listing B.1: Add or delete a specific Vyatta VM Ethernet interface

```
#!/bin/sh
function usage(){
       echo "usage: modify-eth.sh [-a|-d] eth# ipAddress"
       exit 1
}
if [ "$#" -ne 3 ]; then
    usage
else
        case "$1" in
              "-a")
                     CMD="set";
              ;;
              "-d")
                     CMD="delete";
              ;;
              *)
                     usage
              ;;
       esac
       #--- execute
       source /etc/default/vyatta
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper begin
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper "$CMD" interfaces ethernet "$2" address "$3"
       echo "Result from set command="$?
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper commit
       echo "Result from commit command="$?
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper save
fi
```

B.2 Managing Virtual Network Routing

Listing B.2: Script to manage a route between two networks

```
#!/bin/sh
function usage(){
       echo "usage: manage-route.sh [-a|-d] routeAddress destination"
       exit 1
}
if [ "$#" -lt 2 ]; then
       usage
else
        case "$1" in
              "-a")
                    CMD="set protocols static route $2 next-hop $3";
              ;;
              "-d")
                    CMD="delete protocols static route $2";
              ;;
              *)
                     usage
              ;;
       esac
   #--- execute
       source /etc/default/vyatta
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper begin
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper $CMD
       echo "Result from set command="$?
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper commit
       echo "Result from commit command="$?
       /opt/vyatta/sbin/vyatta-cfg-cmd-wrapper save
fi
```

Link-related Enums

Appendix C

XML Schema Definition

C.1 XSD for Resource Description

Listing C.1: The used XSD for describing physical and virtual resources

```
    <?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://xml.netbeans.org/schema/commonTypes"
           xmlns:tns="http://xml.netbeans.org/schema/commonTypes" elementFormDefault="qualified">
    <!-- Enumurations Types-->
    <xs:simpleType name="networkInterfaceEnum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Ethernet"/>
            <xs:enumeration value="gigabitEthernet"/>
            <xs:enumeration value="opticalFiber"/>
            <xs:enumeration value="radio"/>
            <xs:enumeration value="ATM"/>
```

```
            <xs:enumeration value="fameRelay"/>
            <xs:enumeration value="ISDN"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="securityPropSupportedEnum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="confidentiality"/>
            <xs:enumeration value="integrity"/>
            <xs:enumeration value="availability"/>
            <xs:enumeration value="confidentialityAndIntegrity"/>
            <xs:enumeration value="confidentialityAndIntegrityAndAvailability"/>
            <xs:enumeration value="accuracy"/>
            <xs:enumeration value="authenticity"/>
            <xs:enumeration value="utility"/>
            <xs:enumeration value="possession"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="cpuType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Intel"/>
            <xs:enumeration value="AMD"/>
            <xs:enumeration value="Xeon"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ethNumberType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="eth0"/>
            <xs:enumeration value="eth1"/>
            <xs:enumeration value="eth2"/>
            <xs:enumeration value="eth3"/>
            <xs:enumeration value="eth4"/>
            <xs:enumeration value="eth5"/>
        </xs:restriction>
```

```
    </xs:simpleType>
    <xs:simpleType name="osType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Windows"/>
            <xs:enumeration value="Linux"/>
            <xs:enumeration value="Unix"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="hashingTechniqueType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="CyclicRedundancyChecks"/>
            <xs:enumeration value="Checksums"/>
            <xs:enumeration value="Non-cryptographicHhashFunctions"/>
            <xs:enumeration value="CryptographicHashFunctions"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="energySourceType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="renewable"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="memoryType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="DDR3"/>
            <xs:enumeration value="DDR2"/>
            <xs:enumeration value="DDR1"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="netElementStatusType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="nonDeployed" />
            <xs:enumeration value="deployed" />
            <xs:enumeration value="deployedAndAvailable" />
```

```
            <xs:enumeration value="deployedNonAvailable" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="networkSTackType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="TCP/IP" />
            <xs:enumeration value="UDP" />
            <xs:enumeration value="IP/ATM" />
            <xs:enumeration value="IP/Ethernet" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="storageType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SSD" />
            <xs:enumeration value="HDD" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="networkManagementType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SNMP" />
            <xs:enumeration value="CISCO" />
            <xs:enumeration value="IOS XP" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="roleType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="PIP" />
            <xs:enumeration value="VIP" />
            <xs:enumeration value="Broker" />
            <xs:enumeration value="SP" />
            <xs:enumeration value="Customer" />
        </xs:restriction>
    </xs:simpleType>
```

```
<xs:simpleType name="nodeType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="VM" />
        <xs:enumeration value="vSwitch" />
        <xs:enumeration value="vRouter" />
        <xs:enumeration value="PC" />
        <xs:enumeration value="Blade" />
        <xs:enumeration value="Workstation" />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="virEnvironementType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="VMWare" />
        <xs:enumeration value="XEN" />
        <xs:enumeration value="KVM" />
        <xs:enumeration value="OpenVZ" />
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="FunctionalParamType">
    <xs:sequence>
        <xs:element name="NetworkMngmtSWT" type="tns:networkManagementType"/>
        <xs:element name="nbrOfPorts" type="xs:int"/>
        <xs:element name="processingPower" type="xs:string"/>
        <!--Referenced types-->
        <xs:element name="storageParam" type="tns:storageParamType"/>
        <xs:element name="memoryParam" type="tns:memoryParamType"/>
        <xs:element name="cpuParameters" type="tns:cpuParamType"/>
        <xs:element name="osParameters" type="tns:osParamType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="NonFunctionalParamType">
    <xs:sequence>
```

```
        <xs:element name="qosLevelSupported" type="xs:string"/>
        <xs:element name="energyEfficiencyLevel" type="xs:string"/>
        <xs:element name="perOfFailure" type="xs:string"/>
        <xs:element name="cost" type="xs:string"/>
        <!--Referenced types-->
        <xs:element name="energySource" type="tns:energySourceType"/>
        <xs:element name="qosParam" type="tns:qosParamType"/>
        <xs:element name="securityLevelParam" type="tns:securityLevelParamType"/>
        <xs:element name="performanceParam" type="tns:performanceParametersType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="physicalNodeType">
    <xs:complexContent>
        <xs:extension base="tns:networkElementType">
            <xs:sequence>
                <xs:element name="maxNbrOfVirtualNode" type="xs:int"/>
                <xs:element name="substrateNodeGroupID" type="xs:string"/>
                <xs:element name="virEnvironement" type="tns:virEnvironementType"/>
                <xs:element name="nodeType" type="tns:nodeType"/>
                <xs:element name="vendor" type="xs:string"/>
                <xs:element name="model" type="xs:string"/>
                <xs:element name="geoLocation" type="tns:geoLocationType"/>
                <xs:element name="virtualNodes" type="tns:virtualNodes"></xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="virtualNodes">
    <xs:sequence>
        <xs:element name="virtualNode" maxOccurs="unbounded" type="tns:virtualNodeType" />
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="geoLocationType">
    <xs:sequence>
        <xs:element name="country" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="room" type="xs:string"/>
        <xs:element name="rack" type="xs:string"/>
        <xs:element name="panel" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="qosParamType">
    <xs:sequence>
        <xs:element name="avgPacketLoss" type="xs:string"/>
        <xs:element name="avgDelay" type="xs:string"/>
        <xs:element name="avgJitter" type="xs:string"/>
        <xs:element name="avgBitRate" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="performanceParamType">
    <xs:sequence/>
</xs:complexType>
<xs:complexType name="storageParamType">
    <xs:sequence>
        <!--Convert this to add support for attributes-->
        <!-- Define attributes such as measurment units type, etc-->
        <xs:element name="diskSpace" type="xs:string"/>
        <xs:element name="nbrOfUnits" type="xs:int"/>
        <xs:element name="storageType" type="tns:storageType"/>
        <xs:element name="componentInfo" type="tns:componentInfoType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="routingPlatformType">
    <xs:sequence>
```

```
        <xs:element name="name" type="xs:string" />
        <xs:element name="description" type="xs:string"/>
        <xs:element name="version" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="virtualNodeType">
    <xs:sequence>
        <xs:element name="physicalNodeID" type="xs:string"/>
        <xs:element name="providerID" type="xs:string"/>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="description" type="xs:string"/>
        <xs:element name="availability" type="xs:string"/>
        <xs:element name="startTime" type="xs:string"/>
        <xs:element name="networkStack" type="tns:networkSTackType"/>
        <xs:element name="nodeType" type="tns:nodeType"/>
        <xs:element name="virEnvironement" type="tns:virEnvironementType"/>
        <xs:element name="ipAddress" type="xs:string"/>
        <xs:element name="virNetworkInterfaces" type="tns:virNetworkInterfaces" minOccurs="1" />
        <xs:element name="virRoutes" type="tns:virRoutes" minOccurs="1" />
        <xs:element name="routingPlatform" type="tns:routingPlatformType"/>
        <xs:element name="functionalParam" type="tns:FunctionalParamType"/>
        <xs:element name="nonFunctionalParam" type="tns:NonFunctionalParamType"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
</xs:complexType>
<xs:complexType name="virRoutes">
    <xs:sequence>
        <xs:element name="virRoute" type="tns:virRouteType" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="virRouteType">
    <xs:sequence>
        <xs:element name="route" type="xs:string"/>
```

```
        <xs:element name="nextHop" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="virNetworkInterfaces">
    <xs:sequence>
        <xs:element name="virNetworkInterface" type="tns:virtualInterfaceType" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="virtualInterfaceType">
    <xs:sequence>
        <xs:element name="interfaceType" type="tns:networkInterfaceEnum"/>
        <xs:element name="rate" type="xs:string"/>
        <xs:element name="macAddress" type="xs:string"/>
        <xs:element name="ethPortNumber" type="tns:ethNumberType"/>
        <xs:element name="ipAddress" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="providerInfoType">
    <xs:sequence>
        <xs:element name="roleInfo" type="tns:roleInfoType"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
</xs:complexType>
<xs:complexType name="contactInfoType">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="phone" type="xs:string"/>
        <xs:element name="email" type="xs:string" />
        <xs:element name="address" type="tns:addressType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="addressType">
    <xs:sequence>
```

```
        <xs:element name="number" type="xs:int"/>
        <xs:element name="street" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="state" type="xs:string"/>
        <xs:element name="zip" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="roleInfoType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="roleType" type="tns:roleType" />
        <xs:element name="contactInfo" type="tns:contactInfoType" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="networkElementType">
    <xs:sequence>
        <xs:element name="ownerID" type="xs:string"/>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="description" type="xs:string"/>
        <xs:element name="availability" type="xs:string"/>
        <xs:element name="startTime" type="xs:dateTime"/>
        <xs:element name="period" type="xs:string"/>
        <xs:element name="status" type="tns:netElementStatusType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="networkDomainType">
    <xs:sequence>
        <xs:element name="providerName" type="xs:string"/>
        <xs:element name="description" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="administrativeDomainType">
```

```
        <xs:sequence>
            <xs:element name="address" type="tns:addressType"/>
            <xs:element name="description" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="componentInfoType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="vendor" type="xs:string"/>
            <xs:element name="model" type="xs:string"/>
            <xs:element name="partNumber" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="memoryParamType">
        <xs:sequence>
            <xs:element name="size" type="xs:string"/>
            <xs:element name="capacity" type="xs:string"/>
            <xs:element name="memoryType" type="tns:memoryType"/>
            <xs:element name="speed" type="xs:string"/>
            <xs:element name="componentsInfo" type="tns:componentInfoType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="cpuParamType">
        <xs:sequence>
            <xs:element name="cpuType" type="tns:cpuType"/>
            <xs:element name="architecture" type="xs:string"/>
            <xs:element name="nbrOfCores" type="xs:int"/>
            <xs:element name="nbrOfThreads" type="xs:int"/>
            <xs:element name="clockSpeed" type="xs:string" minOccurs="1"/>
            <xs:element name="cahce" type="xs:string" minOccurs="1"/>
            <xs:element name="instructionSet" type="xs:string"/>
            <xs:element name="componentsInfo" type="tns:componentInfoType"/>
        </xs:sequence>
```

```
    </xs:complexType>
    <xs:complexType name="osParamType">
        <xs:sequence>
            <xs:element name="osType" type="tns:osType"/>
            <xs:element name="description" type="xs:string"/>
            <xs:element name="version" type="xs:string"/>
            <xs:element name="vendor" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="securityLevelParamType">
        <xs:sequence/>
    </xs:complexType>
    <xs:complexType name="performanceParametersType">
        <xs:sequence>
            <xs:element name="responseTime" type="xs:string"/>
            <xs:element name="uptime" type="xs:string"/>
            <xs:element name="relaibilityLevel" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```


C.2 XSD for Resource Discovery Requests

Listing C.2: XSD for formulating Resource Discovery requests

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://xml.netbeans.org/schema/discovery"
           xmlns:tns="http://xml.netbeans.org/schema/discovery"
           elementFormDefault="qualified"
           xmlns:cmn="http://xml.netbeans.org/schema/commonTypes">
    <!--Dependencies-->
```

```
<xs:import namespace="http://xml.netbeans.org/schema/commonTypes"
<!--Resource Discovery Request-->
          schemaLocation="commonTypes.xsd"/>
<xs:element name="discoveryRequest">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="selectionParameters" maxOccurs="unbounded" type="tns:selectionParameter" minOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required" />
    </xs:complexType>
</xs:element>
<xs:simpleType name="constraintType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="CPU"/>
        <xs:enumeration value="Memory"/>
        <xs:enumeration value="Storage"/>
        <xs:enumeration value="Link"/>
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="rangeType">
    <xs:sequence>
        <xs:element name="min" type="xs:integer"/>
        <xs:element name="max" type="xs:integer" />
    </xs:sequence>
</xs:complexType>
<!--Constraints used in the selection process-->
<xs:complexType name="selectionConstraint">
    <xs:sequence>
        <xs:element name="constraintOn" type="tns:constraintType" minOccurs="1" />
        <xs:element name="params" type="tns:constraintParams" minOccurs="1" />
        <!-- <xs:element name="range" type="tns:rangeType"/>
        <xs:element name="pattern" type="xs:string"/> -->
    </xs:sequence>
```

```
    </xs:complexType>
    <!--Resource selection parameters-->
    <xs:complexType name="selectionParameter">
        <xs:sequence>
            <xs:element name="vnType" type="cmn:nodeType" minOccurs="1"/>
            <xs:element name="virEnvironement" type="cmn:virEnvironementType" minOccurs="1"/>
            <xs:element name="osType" type="cmn:osType" minOccurs="1"/>
            <xs:element name="interfaceType" type="cmn:networkInterfaceEnum" minOccurs="1"/>
            <xs:element name="networkStack" type="cmn:networkSTackType" minOccurs="1"/>
            <xs:element name="cpuType" type="cmn:cpuType" minOccurs="1"/>
            <xs:element name="quantity" type="xs:integer" minOccurs="1"/>
            <xs:element name="selectionConstraints" type="tns:selectionConstraints" minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="selectionConstraints">
        <xs:sequence>
        <xs:element name="selectionConstraint" type="tns:selectionConstraint" minOccurs="1" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="constraintParams">
        <xs:sequence>
        <xs:element name="param" type="tns:param" minOccurs="1" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="param">
        <xs:attribute name="key" type="xs:string"/>
        <xs:attribute name="value" type="xs:string"/>
    </xs:complexType>
</xs:schema>
```

C.3 XSD for Negotiation Requests

Listing C.3: Schema for resource negotiation requests

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://xml.netbeans.org/schema/requests"
           xmlns:tns="http://xml.netbeans.org/schema/requests"
           elementFormDefault="qualified" xmlns:cmn="http://xml.netbeans.org/schema/commonTypes">
    <!--Dependencies-->
    <xsd:import namespace="http://xml.netbeans.org/schema/commonTypes"
                schemaLocation="commonTypes.xsd"/>
    <xsd:element name="negotiationRequest">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="description" type="xsd:string"/>
                <xsd:element name="price" type="xsd:string"/>
                <xsd:element name="quantity" type="xsd:int"/>
                <xsd:element name="comment" type="xsd:string"/>
                <xsd:element name="pipResponse" type="tns:requestStatusType"/>
                <xsd:element name="vipResponse" type="tns:requestStatusType"/>
                <xsd:element name="requestStatus" type="tns:requestStatusType"/>
                <xsd:element name="virtualNodes" type="cmn:virtualNodes"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="requestStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Accepted" />
            <xsd:enumeration value="Rejected" />
            <xsd:enumeration value="NotProcessed" />
            <xsd:enumeration value="Instantiated" />
```

```
            <xsd:enumeration value="ToBeInstantiated" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

C.4 Resource Description Sample

Listing C.4: A sample of resource description document

```
<?xml version="1.0" encoding="UTF-8"?>
<resourceDescription>
            <virtualNode id="234be927-238f-4abc-bf13-16084f094cc2">
                <physicalNodeID>1ea9bd59-af46-4e93-869c-c29b6c48d3d9</physicalNodeID>
                <providerID>PIP-1</providerID>
                <name>Alpha</name>
                <description>VMWare-based Linux virtual machine</description>
                <availability>yes</availability>
                <startTime>2013/04/14 09:00:24</startTime>
                <networkStack>TCP/IP</networkStack>
                <nodeType>VM</nodeType>
                <virEnvironement>VMWare</virEnvironement>
                <ipAddress>192.168.56.90</ipAddress>
                <virNetworkInterfaces>
                    <virNetworkInterface>
                        <interfaceType>Ethernet</interfaceType>
                        <rate>100MB/sec</rate>
                        <macAddress>24-77-03-5E-30-03</macAddress>
                        <ethPortNumber>eth0</ethPortNumber>
                        <ipAddress>192.168.56.90</ipAddress>
                    </virNetworkInterface>
                    <virNetworkInterface>
                        <interfaceType>Ethernet</interfaceType>
```

```
                <rate>100MB/sec</rate>
                <macAddress>24-77-03-5A-40-03</macAddress>
                <ethPortNumber>eth1</ethPortNumber>
                <ipAddress>192.168.10.10</ipAddress>
            </virNetworkInterface>
        </virNetworkInterfaces>
        <virRoutes>
            <virRoute>
                <route>192.168.20.0</route>
                <nextHop>192.168.56.91</nextHop>
            </virRoute>
        </virRoutes>
        <routingPlatform>
            <name>XORP</name>
            <description>XORP open source routing platform</description>
            <version>1.8.5</version>
        </routingPlatform>
        <functionalParam>
            <NetworkMngmtSWT>SNMP</NetworkMngmtSWT>
            <nbrOfPorts>2</nbrOfPorts>
            <processingPower>200</processingPower>
            <storageParam>
                <diskSpace>40GB</diskSpace>
                <nbrOfUnits>2</nbrOfUnits>
                <storageType>HDD</storageType>
                <componentInfo>
                    <name>Virt Stodage Disk</name>
                    <vendor>Samsung</vendor>
                    <model>6H500F0</model>
                    <partNumber>HTS722020K9A300</partNumber>
                </componentInfo>
            </storageParam>
            <memoryParam>
```

```
                    <size>1024</size>
                    <capacity>1GB</capacity>
                    <memoryType>DDR2</memoryType>
                    <speed>1333</speed>
                    <componentsInfo>
                        <name>DDR2 PC2-5300 memory module</name>
                        <vendor>Kingston</vendor>
                        <model>KVR1333D3E9SK2/8G</model>
                        <partNumber>KVR667D2N5</partNumber>
                    </componentsInfo>
                </memoryParam>
                <cpuParameters>
                    <cpuType>Xeon</cpuType>
                    <architecture>E5-2600 family</architecture>
                    <nbrOfCores>6</nbrOfCores>
                    <nbrOfThreads>12</nbrOfThreads>
                    <clockSpeed>2.3</clockSpeed>
                    <cahce>15MB</cahce>
                    <instructionSet>64-bit</instructionSet>
                    <componentsInfo>
                        <name>Xeon E5-2630-2.3GHz(2.8GHz Turbo Boost)</name>
                        <vendor>Intel</vendor>
                        <model>BX80621E52630</model>
                        <partNumber>BX80621E52630</partNumber>
                    </componentsInfo>
                </cpuParameters>
                <osParameters>
                    <osType>Linux</osType>
                    <description>Debian Linux</description>
                    <version>6</version>
                    <vendor>Brocade</vendor>
                </osParameters>
            </functionalParam>
```

```
            <nonFunctionalParam>
                <qosLevelSupported></qosLevelSupported>
                <energyEfficiencyLevel></energyEfficiencyLevel>
                <perOfFailure>0.01 </perOfFailure>
                <cost>0.99\$/day</cost>
                <energySource>renewable</energySource>
                <qosParam>
                    <avgPacketLoss>1.5</avgPacketLoss>
                    <avgDelay>2ms</avgDelay>
                    <avgJitter></avgJitter>
                    <avgBitRate></avgBitRate>
                </qosParam>
                <securityLevelParam></securityLevelParam>
                <performanceParam>
                    <responseTime>4ms</responseTime>
                    <uptime>16hours</uptime>
                    <relaibilityLevel>II</relaibilityLevel>
                </performanceParam>
            </nonFunctionalParam>
        </virtualNode>
    </virtualNodes>
</resourceDescription>
```

Appendix D

Message Logs of the PIP Subsystem

Listing D.1: Message log generated by the implemented modules involved in the resource publication process

```
15:11:02.597 [INFO] PIPConsole: Starting PIP Console
15:11:52.603 [INFO] PIPResourceHelper: ------------------------------------------------------------
15:11:52.603 [INFO] PIPResourceHelper: Resource publication started...
15:12:03.141 [ERROR] PIPResourceHelper: Proccessing resource file...
15:12:03.142 [ERROR] PIPResourceHelper: file path:/home/sleiman/pip-console/src/descfiles/precision-390_4_routers.xml
15:12:03.243 [INFO] PIPDBResourceManager: Inserting in PIP database...
15:12:03.325 [INFO] PIPDBResourceManager: Resource added successfully to the database...
15:12:05.754 [INFO] PIPResourceHelper: ------------------------------------------------------------
15:12:05.755 [INFO] PIPResourceHelper: Resource publication started...
15:12:06.825 [INFO] PIPDBResourceManager: Retrieving resource description from local DB...
15:12:06.857 [INFO] PIPResourcePublisher: Preparing publication request....
```

```
15:12:07.055 [INFO] PIPResourcePublisher: Sending publication request....
15:12:09.309 [INFO] PIPResourcePublisher: Got broker response!....
15:12:09.309 [INFO] PIPResourcePublisher: Resoruce has been published successfully...
15:12:09.309 [INFO] PIPResourcePublisher: Received HTTP Status code: 201
15:12:09.309 [INFO] PIPDBResourceManager: updating resource's status...
15:12:09.309 [INFO] PIPDBResourceManager: Changing selected resource's status to published...
15:12:09.363 [INFO] PIPDBResourceManager: Resource's status has been changed successfully...
15:12:09.363 [INFO] PIPDBResourceManager: ------------------------------------------------------------
```

Listing D.2: Message log generated during PIP-to-VIP resource negotiation process

```
16:30:09.800 [INFO] PIPConsole: Starting PIP Console
16:30:13.532 [INFO] PIPWSManager: Starting grizzly web server...
Dec 12, 2013 4:30:13 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.17 01/17/2013 03:31 PM'
Dec 12, 2013 4:30:14 PM org.glassfish.grizzly.http.server.NetworkListener start
16:30:14.097 [INFO] PIPWSManager: PIP REST web services with WADL available at http://192.168.10.13:9998/v1/
    application.wadl
INFO: Started listener bound to [192.168.10.13:9998]
Dec 12, 2013 4:30:14 PM org.glassfish.grizzly.http.server.HttpServer start
INFO: [HttpServer] Started.
16:30:56.312 [INFO] PIPNegotiationService: ------------------------------------------------------------
16:30:56.312 [INFO] PIPNegotiationService: Resource negotiation request received...
16:30:56.312 [INFO] PIPNegotiationService: Processing resource negotiation request...
16:30:56.313 [INFO] PIPNegotiationService: Checking if the same request exisits already...
16:30:56.331 [INFO] PIPNegotiationService: Negotiation request exists...
16:30:56.332 [INFO] PIPNegotiationService: Updating existing request...
```

```
16:30:56.501 [INFO] PIPNegotiationService: Negotiation Request status has been updated successfully...
16:30:56.501 [INFO] PIPNegotiationService: Done processing request...
16:30:56.501 [INFO] PIPNegotiationService: ------------------------------------------------------------
16:31:10.984 [INFO] PIPDBResourceManager: Changing negotiation request's status to rejected
16:31:11.049 [INFO] PIPDBResourceManager: Request status has been marked rejected successfully...
16:31:18.652 [INFO] VIPServiceClient: Sending negotiation request to VIP...
16:31:18.653 [INFO] VIPServiceClient: VIP's service URL: http://192.168.10.14:9997/v1/
16:31:18.866 [INFO] PIPDBResourceManager: Changing negotiation request's status to processed...
16:31:18.885 [INFO] PIPDBResourceManager: Request status has been updated successfully...
16:31:51.425 [INFO] PIPNegotiationService: ------------------------------------------------------------
16:31:51.425 [INFO] PIPNegotiationService: Resource negotiation request received...
16:31:51.425 [INFO] PIPNegotiationService: Processing resource negotiation request...
16:31:51.425 [INFO] PIPNegotiationService: Checking if the same request exisits already...
16:31:51.443 [INFO] PIPNegotiationService: Negotiation request exists...
16:31:51.443 [INFO] PIPNegotiationService: Updating existing request...
16:31:51.550 [INFO] PIPNegotiationService: Negotiation Request status has been updated successfully...
16:31:51.550 [INFO] PIPNegotiationService: Done processing request...
16:31:51.550 [INFO] PIPNegotiationService: ------------------------------------------------------------
16:32:00.934 [INFO] PIPDBResourceManager: Changing negotiation request's status to accepted
16:32:01.003 [INFO] PIPDBResourceManager: Request status has been marked accepted successfully...
16:32:04.674 [INFO] VIPServiceClient: Sending negotiation request to VIP...
16:32:04.675 [INFO] VIPServiceClient: VIP's service URL: http://192.168.10.14:9997/v1/
16:32:04.857 [INFO] PIPDBResourceManager: Changing negotiation request's status to processed...
16:32:04.874 [INFO] PIPDBResourceManager: Request status has been updated successfully...
16:32:24.579 [INFO] PIPNegotiationService: ------------------------------------------------------------
```

Listing D.3: Message log of the virtual topology instantiation and configuration process

```
18:29:06.952 [INFO] PIPSubstrateManager: ================================================================
18:29:06.954 [INFO] PIPSubstrateManager: Creating virtual resources and setting up the virtual topology...
18:29:06.955 [INFO] PIPSubstrateManager: ................................................................
18:29:06.955 [INFO] PIPSubstrateManager: Number of resources to instantiate: 1
18:29:06.957 [INFO] PIPSubstrateManager: Connection to host 192.168.56.10 established...
18:29:06.986 [INFO] PIPSubstrateManager: Loging in to host: 192.168.56.10
18:29:07.261 [INFO] PIPSubstrateManager: XAPI Sesscion created successfully
18:29:07.381 [INFO] PIPSubstrateManager: Host lable alpha.encs.concordia.ca
18:29:07.381 [INFO] PIPSubstrateManager: Host UID 6ed32302-b406-4cd8-be96-271dbfedc2b3
18:29:07.381 [INFO] PIPSubstrateManager: Host IP Address 192.168.56.10
18:29:07.381 [INFO] PIPSubstrateManager: Editionfree
18:29:07.381 [INFO] PIPSubstrateManager: Free memory 1762967552
18:29:07.381 [INFO] PIPSubstrateManager: ----------------------------
18:29:07.382 [INFO] PIPSubstrateManager: Creating virtual machines:
18:29:07.382 [INFO] PIPSubstrateManager: ----------------------------
18:29:07.382 [INFO] PIPSubstrateManager:  Creating VM => Alpha ...
18:29:07.831 [INFO] PIPSubstrateManager:  We're creating: Alpha VM from VyattaVMTemplate
18:29:16.201 [INFO] PIPSubstrateManager:  VM Alpha created successfully...
18:29:16.201 [INFO] PIPSubstrateManager:  Configuring Alpha's network settings...
18:29:16.230 [INFO] PIPSSHRemoteCmd:   Opening SSH connection...
18:29:16.230 [INFO] PIPSSHRemoteCmd:   Please wait, this might take some time...
18:29:16.230 [INFO] PIPSSHRemoteCmd:   Waiting for SSH server to go online...
18:30:22.498 [INFO] PIPSSHRemoteCmd:   The VM is reachable now! ...
```

```
18:30:22.499 [INFO] PIPSSHRemoteCmd:   Executing command: sh /home/vyatta/script/modify-eth.sh -a eth1
    192.168.10.10/24 ...
18:30:22.526 [INFO] PIPSSHRemoteCmd:   Here is some information about the remote host:
18:30:22.656 [INFO] PIPSSHRemoteCmd:   Result from set command=0
18:30:23.387 [INFO] PIPSSHRemoteCmd:   Result from commit command=0
18:30:23.442 [INFO] PIPSSHRemoteCmd:   Saving configuration to '/opt/vyatta/etc/config/config.boot'...
18:30:23.475 [INFO] PIPSSHRemoteCmd:   Done
18:30:23.477 [INFO] PIPSSHRemoteCmd:   ExitCode: 0
18:30:23.477 [INFO] PIPSSHRemoteCmd:   Closing SSH session now...
18:30:23.478 [INFO] PIPSSHRemoteCmd:   Executing command: sh /home/vyatta/script/modify-eth.sh -a eth0
    192.168.56.90/24 ...
18:30:23.517 [INFO] PIPSSHRemoteCmd:   Here is some information about the remote host:
18:30:23.570 [INFO] PIPSSHRemoteCmd:   Result from set command=0
18:30:24.203 [INFO] PIPSSHRemoteCmd:   Result from commit command=0
18:30:24.257 [INFO] PIPSSHRemoteCmd:   Saving configuration to '/opt/vyatta/etc/config/config.boot'...
18:30:24.291 [INFO] PIPSSHRemoteCmd:   Done
18:30:24.292 [INFO] PIPSSHRemoteCmd:   ExitCode: 0
18:30:24.292 [INFO] PIPSSHRemoteCmd:   Closing SSH session now...
18:30:24.293 [INFO] PIPSSHRemoteCmd:   Executing command: sh /home/vyatta/script/modify-eth.sh -d eth5
    192.168.56.199/24 ...
18:30:24.327 [INFO] PIPSSHRemoteCmd:   Ethernet interface deleted...
18:30:25.328 [INFO] PIPSSHRemoteCmd:   Closing SSH session now...
18:30:25.329 [INFO] PIPSSHRemoteCmd:   Closing SSH connection...
18:30:25.330 [INFO] PIPSubstrateManager:  Finished creating virtual machines!
18:30:25.331 [INFO] PIPSubstrateManager: ----------------------------------------
18:30:25.331 [INFO] PIPSubstrateManager: Configuring and setting up virtual topology...
```

```
18:30:25.332 [INFO] PIPSubstrateManager: ------------------------------------------
18:30:25.332 [INFO] PIPSSHRemoteCmd:   Opening SSH connection...
18:30:25.332 [INFO] PIPSSHRemoteCmd:   Please wait, this might take some time...
18:30:25.332 [INFO] PIPSSHRemoteCmd:   Waiting for SSH server to go online...
18:30:25.459 [INFO] PIPSSHRemoteCmd:   The VM is reachable now! ...
18:30:25.460 [INFO] PIPSSHRemoteCmd:   Executing command: sh /home/vyatta/script/manage-route.sh -a 192.168.20.0/24
    192.168.56.91 ...
18:30:25.469 [INFO] PIPSSHRemoteCmd:   Here is some information about the remote host:
18:30:25.568 [INFO] PIPSSHRemoteCmd:   Result from set command=0
18:30:26.146 [INFO] PIPSSHRemoteCmd:   Result from commit command=0
18:30:26.201 [INFO] PIPSSHRemoteCmd:   Saving configuration to '/opt/vyatta/etc/config/config.boot'...
18:30:26.234 [INFO] PIPSSHRemoteCmd:   Done
18:30:26.235 [INFO] PIPSSHRemoteCmd:   ExitCode: 0
18:30:26.235 [INFO] PIPSSHRemoteCmd:   Closing SSH session now...
18:30:26.236 [INFO] PIPSSHRemoteCmd:   Closing SSH connection...
18:30:26.237 [INFO] PIPSubstrateManager: Topology instantiated successfully...
18:30:26.237 [INFO] PIPSubstrateManager: ------------------------------------------------------------
```

Listing D.4: Message log of resource publication scalability tests

```
16:49:26.255 [INFO] PubRequestGenerator: ------------------------------------------------
16:49:26.258 [INFO] PubRequestGenerator: Generating resource publication has been started...
16:49:29.905 [INFO] PubRequestGenerator:
 Run number #1:
        Start time:  Mon Jun 20 03:26:55 EDT 2231
        Total elapsed time in milliseconds: 3643 ms
        Total published resources : 20
16:49:32.933 [INFO] PubRequestGenerator:
 Run number #2:
        Start time:  Mon Aug 01 08:32:10 EDT 2231
        Total elapsed time in milliseconds: 3028 ms
        Total published resources : 20
16:49:35.778 [INFO] PubRequestGenerator:
 Run number #3:
        Start time:  Mon Sep 05 09:48:48 EDT 2231
        Total elapsed time in milliseconds: 2844 ms
        Total published resources : 20
16:49:38.574 [INFO] PubRequestGenerator:
 Run number #4:
        Start time:  Sat Oct 08 08:05:17 EDT 2231
        Total elapsed time in milliseconds: 2794 ms
        Total published resources : 20
16:49:41.391 [INFO] PubRequestGenerator:
 Run number #5:
```

```
        Start time:  Wed Nov 09 15:28:23 EST 2231
        Total elapsed time in milliseconds: 2816 ms
        Total published resources : 20
Average time: 3025 ms
16:49:41.391 [INFO] PubRequestGenerator: Finished genrating resource publication request...
16:49:41.391 [INFO] PubRequestGenerator: ------------------------------------------------
```

Appendix E

Message Logs of the VIP Subsystem

Listing E.1: Resource discovery message log

```
15:29:13.358 [INFO] VIPConsole: Loading VIP console settings...
15:29:13.363 [INFO] VIPConsole: Console settings loaded...
15:29:17.385 [INFO] VIPResourceDiscovery: ------------------------------------------------------------
15:29:17.385 [INFO] VIPResourceDiscovery: Resource discovery started...
15:29:17.386 [INFO] VIPResourceDiscovery: Loading discovery request document...
15:29:17.792 [INFO] VIPResourceDiscovery: Sending discovery request...
15:29:18.372 [INFO] VIPResourceDiscovery: Got server response!....
15:29:18.466 [INFO] VIPResourceDiscovery: Discovery request has been processed successfully...
15:29:18.466 [INFO] VIPResourceDiscovery: HTTP Status code: 200
15:29:18.466 [INFO] VIPResourceDiscovery: Number of discovered resources: 3
15:29:18.466 [INFO] VIPResourceDiscovery: ------------------------------------------------------------
```

Listing E.2: VIP resource negotiation message log

```
16:30:30.607 [INFO] VIPConsole: Loading VIP console settings...
16:30:30.612 [INFO] VIPConsole: Console settings loaded...
16:30:34.428 [INFO] VIPWSManager: Starting grizzly web server...
Dec 12, 2013 4:30:34 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.17 01/17/2013 03:31 PM'
Dec 12, 2013 4:30:35 PM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [192.168.10.14:9997]
Dec 12, 2013 4:30:35 PM org.glassfish.grizzly.http.server.HttpServer start
16:30:35.044 [INFO] VIPWSManager: VIP REST web services started with WADL available at http://192.168.10.14:9997/v1/
    application.wadl
INFO: [HttpServer] Started.
16:30:56.076 [INFO] PIPServiceClient: ------------------------------------------------------------
16:30:56.076 [INFO] PIPServiceClient: Sending negotiation request to PIP...
16:30:56.076 [INFO] PIPServiceClient: PIP service's URL: http://192.168.10.13:9998/v1/
16:30:56.506 [INFO] PIPServiceClient: HTTP status code: 200
16:31:18.731 [INFO] VIPNegotiationService: A negotiation request has been received...
16:31:18.731 [INFO] VIPNegotiationService: Checking if the request exists already...
16:31:18.748 [INFO] VIPNegotiationService: Request exists...
16:31:18.749 [INFO] VIPNegotiationDao: Updating request...
16:31:18.864 [INFO] VIPNegotiationDao: Request has been updated successfully...
16:31:18.865 [INFO] VIPNegotiationService: Negotiation request processed successfully...
16:31:38.399 [INFO] VIPRequestNegoManager: Updating request...
16:31:39.603 [INFO] VIPRequestHelper: Changing request's status to accepted...
16:31:51.369 [INFO] PIPServiceClient: ------------------------------------------------------------
```

```
16:31:51.369 [INFO] PIPServiceClient: Sending negotiation request to PIP...
16:31:51.370 [INFO] PIPServiceClient: PIP service's URL: http://192.168.10.13:9998/v1/
16:31:51.551 [INFO] PIPServiceClient: HTTP status code: 200
16:32:04.734 [INFO] VIPNegotiationService: A negotiation request has been received...
16:32:04.734 [INFO] VIPNegotiationService: Checking if the request exists already...
16:32:04.751 [INFO] VIPNegotiationService: Request exists...
16:32:04.751 [INFO] VIPNegotiationDao: Updating request...
16:32:04.856 [INFO] VIPNegotiationDao: Request has been updated successfully...
16:32:04.857 [INFO] VIPNegotiationService: Negotiation request processed successfully...
16:32:23.363 [INFO] VIPRequestNegoManager: Changing request status to be instantiated...
16:32:23.363 [INFO] VIPRequestNegoManager: Sending instantiation request to PIP...
16:32:24.528 [INFO] PIPServiceClient: ------------------------------------------------------------
16:32:24.528 [INFO] PIPServiceClient: Sending negotiation request to PIP...
16:32:24.529 [INFO] PIPServiceClient: PIP service's URL: http://192.168.10.13:9998/v1/
16:32:24.734 [INFO] PIPServiceClient: HTTP status code: 200
```

Listing E.3: Message log of resource discovery scalability tests

```
16:46:25.054 [INFO] DiscoveryRequstGenerator: ------------------------------------------------
16:46:25.056 [INFO] DiscoveryRequstGenerator: Generation of resource discovery requests has been started...
16:46:25.969 [INFO] DiscoveryRequstGenerator:
 Run number #1:
 Discovery statistics:
        Start time:  Wed Sep 21 22:23:37 EDT 2225
        Total elapsed time in milliseconds: 909 ms
        Total discoverd resources : 10
16:46:26.494 [INFO] DiscoveryRequstGenerator:
 Run number #2:
 Discovery statistics:
        Start time:  Sun Oct 02 12:02:47 EDT 2225
        Total elapsed time in milliseconds: 524 ms
        Total discoverd resources : 10
16:46:26.987 [INFO] DiscoveryRequstGenerator:
 Run number #3:
 Discovery statistics:
        Start time:  Sat Oct 08 13:48:59 EDT 2225
        Total elapsed time in milliseconds: 492 ms
        Total discoverd resources : 10
16:46:27.498 [INFO] DiscoveryRequstGenerator:
 Run number #4:
 Discovery statistics:
        Start time:  Fri Oct 14 06:35:16 EDT 2225
```

```
        Total elapsed time in milliseconds: 510 ms
        Total discoverd resources : 10
16:46:27.985 [INFO] DiscoveryRequstGenerator:
 Run number #5:
 Discovery statistics:
        Start time:  Thu Oct 20 04:32:07 EDT 2225
        Total elapsed time in milliseconds: 486 ms
        Total discoverd resources : 10
16:46:27.985 [INFO] DiscoveryRequstGenerator: Average time: 584 ms
16:46:27.985 [INFO] DiscoveryRequstGenerator: Finished request generation...
16:46:27.985 [INFO] DiscoveryRequstGenerator: ------------------------------------------------
```

Appendix F

Broker Components Message Logs

Listing F.1: Broker's resource publication message logs

```
15:27:18.437 [INFO] BrokerConsole: Starting Broker console ...
15:27:18.439 [INFO] BrokerConsole: Setting Nimbus look and fell ...
15:27:20.374 [INFO] BrokerServiceStarter: Starting grizzly web server!...
Dec 12, 2013 3:27:20 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.17 01/17/2013 03:31 PM'
Dec 12, 2013 3:27:20 PM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [192.168.10.12:9995]
Dec 12, 2013 3:27:20 PM org.glassfish.grizzly.http.server.HttpServer start
15:27:20.963 [INFO] BrokerServiceStarter: Jersey app started with WADL available at http://192.168.10.12:9995/v1/
    application.wadl
INFO: [HttpServer] Started.
15:27:38.743 [INFO] PublicationService: ------------------------------------------------------------
```

```
15:27:38.743 [INFO] PublicationService: Resource publication request received...
15:27:38.938 [INFO] PublicationDao: Adding new resource to the DB...
15:27:38.938 [INFO] PublicationDao: Processing resource with ID: bf853b59-1f5f-485d-bacb-2c6f16ba9b9a
15:27:39.022 [INFO] PublicationDao: Adding new resource to the DB...
15:27:39.022 [INFO] PublicationDao: Processing resource with ID: 8f2fd41a-f692-4b35-8f9a-7db735667d17
15:27:39.090 [INFO] PublicationDao: Adding new resource to the DB...
15:27:39.090 [INFO] PublicationDao: Processing resource with ID: 94a8f32a-ddff-4992-b859-36cab710252f
15:27:39.155 [INFO] DaoController: Checking whether the resource exists in the DB...
15:27:39.156 [INFO] DaoController: Resource does not exist...
15:27:39.245 [INFO] DaoController: Resource created successfully in the DB...
15:27:39.245 [INFO] PublicationService: Resource publication done...
15:27:39.245 [INFO] PublicationService: ------------------------------------------------------------
```


Listing F.2: Broker's resource discovery logs

```
 15:29:04.651 [INFO] BrokerConsole: Starting Broker console ...
15:29:04.653 [INFO] BrokerConsole: Setting Nimbus look and fell ...
15:29:06.485 [INFO] BrokerServiceStarter: Starting grizzly web server!...
Dec 12, 2013 3:29:06 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.17 01/17/2013 03:31 PM'
Dec 12, 2013 3:29:07 PM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [192.168.10.12:9995]
15:29:07.071 [INFO] BrokerServiceStarter: Jersey app started with WADL available at http://192.168.10.12:9995/v1/
    application.wadl
Dec 12, 2013 3:29:07 PM org.glassfish.grizzly.http.server.HttpServer start
INFO: [HttpServer] Started.
15:29:18.007 [INFO] DiscoveryService: ------------------------------------------------------------
```

```
15:29:18.007 [INFO] DiscoveryService: Received a discovery request...
15:29:18.007 [INFO] ResourceDiscoveryDao: Processing resource discovery request...
15:29:18.008 [INFO] ResourceDiscoveryDao: Loading available resources' descirption...
15:29:18.008 [INFO] ResourceDiscoveryDao: Finding resource by type...
15:29:18.297 [INFO] ResourceDiscoveryDao: Resource selection algorithm has been started...
15:29:18.298 [INFO] ResourceSelection: Processing selection parameters on resource [52a6ff62-fd26-4ee3-be65-060
    b82140ccb]...
15:29:18.298 [INFO] ResourceSelection: Processing selection constraints on resource [52a6ff62-fd26-4ee3-be65-060
    b82140ccb]...
15:29:18.298 [INFO] ResourceSelection: Evaluating memory constraints...
15:29:18.298 [INFO] ResourceSelection: Evaluating CPU constraints...
15:29:18.298 [INFO] ResourceSelection: Processing selection parameters on resource [3a4d2bdc-e2f8-41c1-a565-
    f606eb337a56]...
15:29:18.298 [INFO] ResourceSelection: Processing selection constraints on resource [3a4d2bdc-e2f8-41c1-a565-
    f606eb337a56]...
15:29:18.298 [INFO] ResourceSelection: Evaluating memory constraints...
15:29:18.298 [INFO] ResourceSelection: Evaluating CPU constraints...
15:29:18.299 [INFO] ResourceSelection: Processing selection parameters on resource [11432e48-186d-44b7-bd7d-
    e80e2f801e1d]...
15:29:18.299 [INFO] ResourceSelection: Processing selection constraints on resource [11432e48-186d-44b7-bd7d-
    e80e2f801e1d]...
15:29:18.299 [INFO] ResourceSelection: Evaluating memory constraints...
15:29:18.299 [INFO] ResourceSelection: Evaluating CPU constraints...
15:29:18.299 [INFO] DiscoveryService: Resource discovery done...
15:29:18.299 [INFO] DiscoveryService: ------------------------------------------------------------
```