Rabah et al. Journal of Cloud Computing: Advances, Systems and Applications (2015) 4:3
DOI 10.1186/s13677-015-0029-5

Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

A service oriented broker-based approach for dynamic resource discovery in virtual networks

Sleiman Rabah[1], May El Barachi[2*], Nadjia Kara[3], Rachida Dssouli[4] and Joey Paquet[5]

Abstract

In the past few years, the concept of network virtualization has received significant attention from industry and research fora. This concept applies virtualization to networking infrastructures by enabling the dynamic creation of several co-existing logical network instances (or virtual networks) over a shared physical network infrastructure (or substrate network). Due to the potential it offers in terms of diversifying existing networks and ensuring the co-existence of heterogeneous network architectures on top of shared substrates, network virtualization is often considered as an enabler of a polymorphic Internet and a cornerstone of the future Internet architecture. One of the challenges associated with the network virtualization concept is the description, publication, and discovery of virtual resources that can be composed to form virtual networks. To achieve those tasks, there is a need for an expressive information model facilitating information representation and sharing, as well as an efficient resource publication and discovery framework. In this paper, we propose a service oriented, broker-based framework for virtual resource description, publication, and discovery. This framework relies on a novel service-oriented hierarchical business model and an expressive information model for resources/services description. The detailed framework's architecture is presented, and its operation is illustrated using a REST-based content distribution scenario. Furthermore, a proof-of-concept prototype implementation realized using various technologies/tools (e.g. Jersey, JAXB, PostgreSQL, and Xen cloud platform) is presented along with a detailed performance analysis of the system. When compared to existing virtual resource discovery frameworks, our broker-based virtual resource discovery framework offers signification performance improvements of the virtual resources' discovery operation, in terms of response time (92.8% improvement) and incurred network load (77.3% improvement), when dealing with multiple resource providers. Furthermore, relying on a broker as intermediary role simplifies the resources' discovery and selection operations, and improves the overall efficiency of the virtual network embedding process.

Keywords: Network virtualization; Dynamic resource discovery; Information modeling; REST; Xen cloud

Introduction

Network virtualization is an emerging concept that applies virtualization to networking infrastructures and promotes a "network-as-a-service" model, in which a dynamic pool of virtualized networking resources can be leased and released on demand. The basic idea behind network virtualization consists in the dynamic creation of several co-existing logical network instances (or virtual networks) over a shared physical network infrastructure (or substrate network) [1]. Offering full administrative control and customization capabilities, virtual networks can be built according to different design criteria and operated as service-tailored networks [2].

Recently, network virtualization has received a lot of attention from industry and research fora, as it represents a promising way to diversify existing networks and ensure the co-existence of heterogeneous network architectures on top of shared substrates [3-5]. In addition, network virtualization enables the emergence of new actors and business roles to offer on-demand virtual networks (VNs), customized for particular service and user requirements.

In the Internet domain, network virtualization is considered as a promising solution for the "Internet ossification" problem – A condition by which the sheer size and scope of the Internet architecture renders the introduction

* Correspondence: may.elbarachi@zu.ac.ae
[2]College of Technological Innovation, Zayed University, Khalifa City B, P.O. Box 144534, Abu Dhabi, United Arab Emirates
Full list of author information is available at the end of the article

and deployment of new technologies very difficult due to the high cost of migration and the difficulty of achieving wide consensus among the many involved stakeholders [6]. By enabling a logical segmentation of the physical Internet infrastructure and the co-existence of heterogeneous virtual networking architectures on top of it, network virtualization is often seen as a cornerstone of the future Internet architecture [3,6].

In order to provide a clean separation of services and infrastructures, the network virtualization concept decouples the traditional Internet Service Provider (ISP) role into two main roles: the Infrastructure Provider (InP) role owning/managing the physical infrastructure and partitioning its resources into isolated slices using some virtualization technology; and the Service Provider (SP) role relying on the infrastructure to offer end-user services. Furthermore, the Virtual Network Provider (VNP) is introduced as a third intermediary role responsible for discovering and aggregating virtual resources from one or multiple InPs in order to instantiate virtual networks (VNs) satisfying customers' requests [1].

Forming the initial steps in the virtual network embedding process, resource description, publication, and discovery are critical phases that aim at assisting the VNP in identifying potential InPs that possess the resources/services needed to satisfy a certain VN request (request coming from a client to instantiate a VN). While resource description focuses on the representation of the functional (i.e. static) and non-functional (i.e. dynamic) attributes of resources/services offered by different providers; resource publication enables the advertisement of this information; and resource discovery enables the searching and finding of resource candidates that comply with the requirements specified in the VN request.

Despite their importance in the VN embedding (or composition) process, little work has been done on resource description, publication, and discovery in virtual networking environments. The solutions proposed so far [7-10] rely on coarse-grained inexpressive information models, and require the interaction between a VNP and a potentially large number of candidate resource providers to gather the necessary information needed for resource selection. In a large scale virtual networking environment in which many InPs offer virtualized resources for lease, such solutions may introduce complexity and result in excessive delays and communication overhead between a VNP and a large number of potential InP candidates – thus impacting the overall efficiency of the VN embedding process. Furthermore, none of the proposed architectures has been fully implemented.

In this work, we propose a service-oriented broker-based framework for resource description, publication, and discovery in virtual networking environments. Our framework promotes the idea of "network-as-a-service" by defining different levels of services to which networking resources are mapped. It relies on a novel service-oriented hierarchical business model [11] that introduces the broker as intermediary role, and proposes the concept of vertical hierarchy between VNPs, as well as the concept of service building block and service reuse and composition. At the heart of the framework lies an expressive information model [12] enabling the representation of physical/virtual resources and services, as well as the mapping between them, in addition to modeling the relations between multiple business roles and their association to resources and services offered. Unlike other approaches, our proposed framework aims at offering an efficient dynamic resource discovery solution enabling the seamless interaction and collaboration between various roles, while coping with the complexity of managing and organizing large numbers of virtualized resources/services.

The rest of the paper is organized as follows: Section 2 gives an overview of the network virtualization concept and presents a review of related work. In Section 3, our proposed approach for virtual resources' description and discovery is discussed in detail, including the business and information models on which it relies, the architectural framework design, the architectural components and interfaces, as well as a REST-based secure content distribution scenario illustrating the system's operation. This is followed by a presentation of the prototype implementation and performance measurements, in Section 4. The last section ends the paper with our conclusions.

Background and related work

The network virtualization concept

A virtual networking environment can be seen as a dynamic and collaborative environment, in which a large pool of virtualized networking resources can be offered and leased on demand. In such an environment, a number of logical network instances (virtual networks) co-exist over a shared physical network infrastructure. A virtual network essentially consists of a set of virtual nodes connected by virtual links, and forming a virtual topology. In this topology, each virtual node (guest) is hosted on a certain physical node (host) and each virtual link is established over a physical path. In this environment, each virtual network is managed/operated by a single entity, and virtual networks are logically isolated from each other.

From an architectural perspective, network virtualization promotes several design goals [1], the most prominent ones being: the *coexistence* of multiple VNs (operated by different providers) within the same environment; *recursion and inheritance* between VNs allowing the nesting/creation of a VN on top of another VN (thus forming a hierarchy); *flexibility* by allowing a provider to implement an arbitrary network topology, routing and forwarding functionalities and customized control protocols in a VN;

manageability allowing a provider to have full administrative control over a VN; *isolation* between co-existing VNs to improve fault-tolerance, security and privacy; and *heterogeneity* of VNs as well as the physical infrastructures on which they rely.

Three main business models were proposed for Network Virtualization Environments (NVEs) [5,11-13]. The initial model proposed in [13] decouples the traditional Internet Service Provider role into two roles: the role of infrastructure provider (InP) managing the physical infrastructure; and the role of service provider (SP) creating VNs by aggregating resources from multiple infrastructure providers and offering end-to-end services. The second model, which has been proposed in the 4Ward project [5], refines the first model by defining four roles: physical infrastructure provider (corresponding to the role of infrastructure provider in the first model); VNP responsible of finding and composing an adequate set of virtual resources from one or more infrastructure providers into an empty virtual topology; virtual network operator (VNO) that deploys different protocols over the VN topology and is responsible for the control and management of the VN; and service provider using the VN to offer end-to-end services. The third model was proposed by us in [11], as a refinement to the previous two models. In this model, we put the emphasis on the notion of services by defining different levels of services that could be offered by networking resources, and introduced the broker role as an additional role needed to enable the collaboration between various entities for service provisioning. Furthermore, we introduced the notion of hierarchy between VNs and virtual infrastructure providers, as well as the idea of service building blocks and service composition.

The virtual network embedding process

As shown in Figure 1, the virtual network embedding process consists of multiple phases and involves the collaboration between different roles. We describe the different phases as follows:

- **Resource Description:** Prior to any VN provisioning operation, physical/virtual infrastructure providers need to describe their available resources. This step relies on information models that enable the description of the functional attributes (i.e. static parameters such as node/link type, OS) and the non-functional attributes (i.e. dynamic parameters such as available capacity/bandwidth) of available resources and services.
- **Resource Publication and Discovery:** Once resources are described, their related information is advertised and dynamically discovered by different entities wishing to make use of those resources. According to the literature, the description of the resources may be registered in public discovery frameworks, or repositories, so that they can be discovered by VN requestors. As for the discovery process, it consists of searching and finding resource candidates that comply with the requirements specified in the VN request. In [14], two aspects of resource discovery have been identified: resource matching and resource splitting. The matching mechanism identifies resources that meet a set of functional and non-functional attributes using clustering techniques and similarity based matching algorithms [15,16]. The splitting mechanism divides virtual network requests into sub-requests that can be handled by different physical resource providers.
- **Resource Selection:** Taking as input the list of resource candidates identified by the resource discovery process, the resource selection process aims at selecting the best candidates that satisfy the requirements specified in the VN request. Depending on the sophistication of the selection approach, multiple operations may be used to obtain the best resources candidates, including filtering, aggregating, ranking and multi-objective-based selection [17]. The selection of the best resources while maximizing the number of allocated virtual networks over a shared physical infrastructure is recognized as an NP-hard problem [18]. Therefore, several approaches have been proposed to tackle this issue, including: exact formulation [19]; heuristics-based [20]; and economic-based models [21].
- **Resource Negotiation:** Resource negotiation is considered as an important process that enables a service provider to negotiate with multiple infrastructure providers (offering similar resources), in order to select the best one. This involves the negotiation of the resources' cost, allocated capabilities over the Service Level Agreement (SLA) period, and the related quality of service scheme.
- **Resource Allocation and Mapping:** Resource allocation or resource provisioning [14] is the process of mapping/binding virtual resources to physical resources (such as nodes and links). Resource allocation is performed by a physical infrastructure provider, and mainly consists of virtual nodes and virtual links creation, configuration and setup, thus resulting in virtual topologies instantiation.
- **Dynamic Resource Management:** Once allocated, virtual networks may be subject to dynamic variations, such as changes in service demands, traffic loads, and resource conditions. To cope with such variations, dynamic resource management strategies are needed. Such strategies may include virtual nodes' migration and dynamic topologies' adaptation.

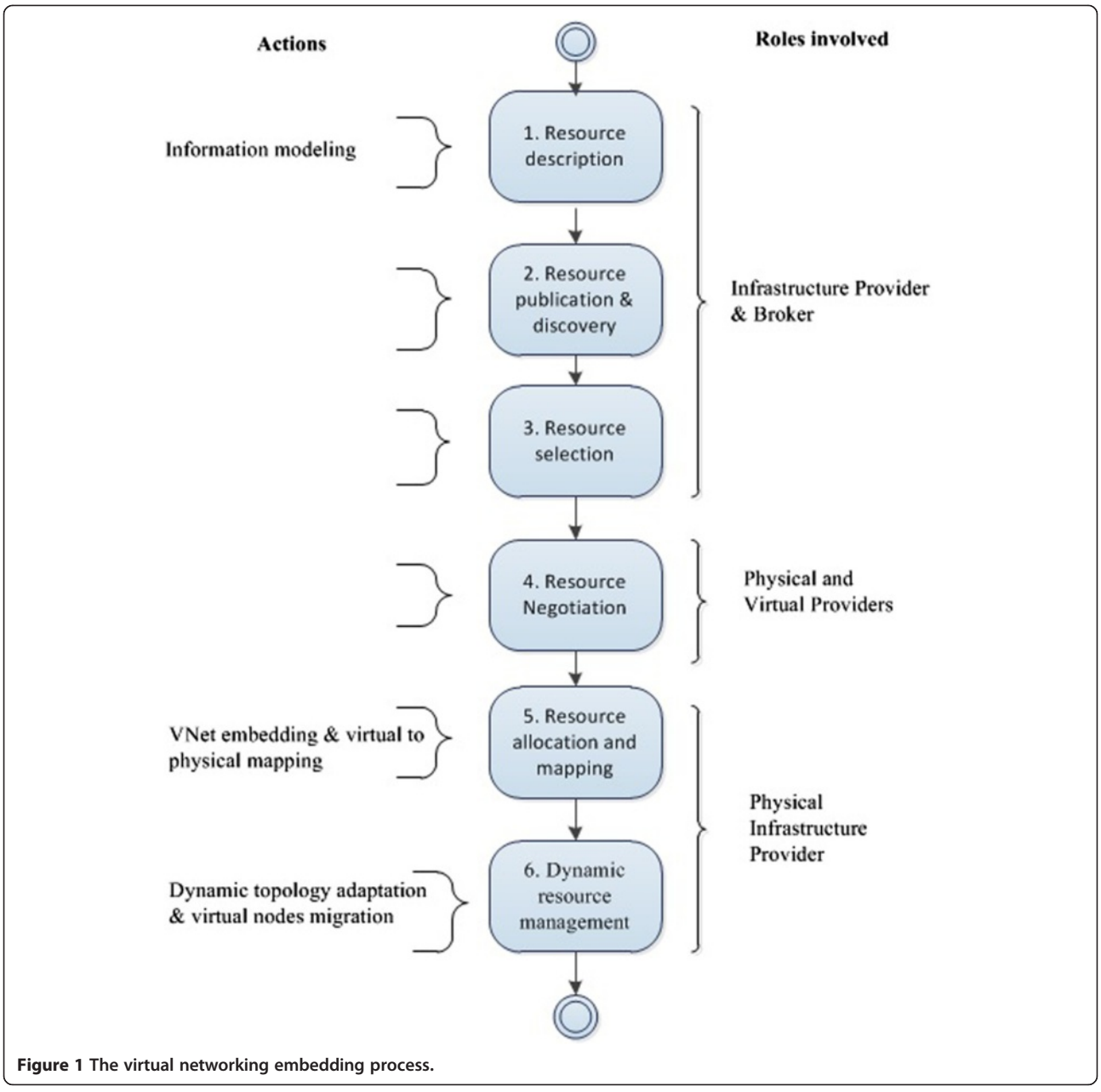

Figure 1 The virtual networking embedding process.

Resource description and discovery in virtual networking environments

In a federated virtual networking environment (where similar resources could be offered by many providers), resource description and discovery are critical processes that aim at assisting the VNP in identifying potential InPs that possess the resources/services needed to satisfy a certain VN request. Despite their importance in the VN embedding process, little work has been done on resource description and discovery in virtual networking environments.

In [7], a virtual resource description and discovery framework has been proposed for the 4WARD model. As part of this framework, a UML-based information model is proposed to describe resources. In this model, the network element is considered as the basic building component having functional and non-functional attributes. Furthermore, this framework uses a conceptual clustering technique to arrange resources' information into a tree structure (dendrogram) that facilitates the matching and selection processes. In this case, only resources' functional attributes are advertised and stored in external repositories to be used for discovery purposes, whereas non-functional attributes are updated and kept in local repositories to be used during selection and binding phases. A distributed peer-to-peer architecture

is proposed for the realization of the proposed resource discovery framework.

Building on the solution proposed in [7], the authors in [8] propose a similar framework that relies on a hierarchical clustering architecture for virtual resource organization and discovery. In this architecture, local management nodes are used to store static (functional) attributes and arranging them into conceptual clusters called Micro Clusters (MiCs) at the InP level. The Cluster Index Server groups/organizes the MiCs of multiple InPs having the same root attribute, thus resulting in a Macro Cluster (MaC). Time variant attributes such as the residual capacity of a substrate link is stored in the management node. This framework aims at reducing the search range and cost as well as enhancing the efficiency of the resource discovery process.

To benefit from WSDL's support for dynamic update services, authors in [9] proposed a WSDL-based VN resource provisioning framework. 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 to select the candidate resources, a search in all the UDDI registries is required. In this framework, the UDDI registries parse the information contained in WSDL and use the greedy and shortest path algorithms to retrieve the necessary information.

Aiming at enhancing the efficiency of the selection process by considering dynamic attributes during the discovery phase, the Aggregation-based Discovery for Virtual Network Environments (ADVNE) is proposed in [10]. In order to minimize the continuous monitoring overhead of dynamic attributes, the authors propose to calculate aggregated values of the monitored attributes instead. In this approach, each InP has a monitoring agent that monitors and calculates the aggregated values of two dynamic attributes (nodes' available CPU and links' available bandwidth). Furthermore, InPs publish their resources' static attributes in VNPs managed repositories. Later on, a VNP wishing to perform resource discovery will use its discovery module to retrieve the needed static attributes (from the VNP's repository) and dynamic attributes (from monitoring agents at InPs level). This information will be used to conduct an initial filtering of candidate InPs and provide the resulting list as input to the selection and binding processes.

Despite the merits of the solutions proposed, they all suffer from some drawbacks. In terms of virtual resources' description, the main information model proposed in [7] and adopted in [8,10] provides minimal description of static/dynamic attributes, and lacks the expressiveness needed to describe other important aspects, such as describing a virtual network as a whole, virtual-to-physical mapping, network services description, and modeling of the relationships between roles and resources/services. This lack of expressiveness and coarse-grained approach will limit the ability to accurately describe VN requests and virtual resources/services, thus impacting the selection likelihood and the VN embedding efficiency.

In terms of resource discovery architecture, all the solutions proposed follow the 4WARD business model, which does not define a broker role acting as intermediary between the other roles. As a result, a VNP must gather information from multiple InPs (either through communication with local monitoring agents [10], or distributed repositories [7,9], or cluster heads [8]) before initiating the resource selection process. In a large scale virtual networking environment in which many InPs offer virtualized resources for lease, such approach may introduce complexity and result in excessive delays and communication overhead between a VNP and a large number of potential InP candidates – thus impacting the overall efficiency of the VN embedding process. Furthermore, none of the proposed architectures has been fully implemented - the validation focusing mainly on evaluating the performance of the matching and selection algorithms.

In the coming section, we propose a service-oriented broker based framework for virtual resource description and discovery in virtual networking environments. Our framework relies on a novel service-oriented hierarchical business model as well as an expressive information model enabling the representation of physical/virtual resources and services.

Virtual resources' description and discovery approach

In this section, we first give an overview of the virtual networking business model [11] and the information model [12] that form the basis of this work. Then, we present a novel architecture for the publication and dynamic discovery of resources (PDDR) in network virtualization environment (NVE). Throughout this paper, we use the term resource to refer to a computational/network resource (physical or virtual), network service or role information.

Proposed virtual networking business model

Figure 2 depicts the business model we proposed in [11] for virtualized networking environments. Four levels of service are defined as part of our model, namely: *Essential service*s constituting mandatory services needed for the basic operation of the network (i.e. 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

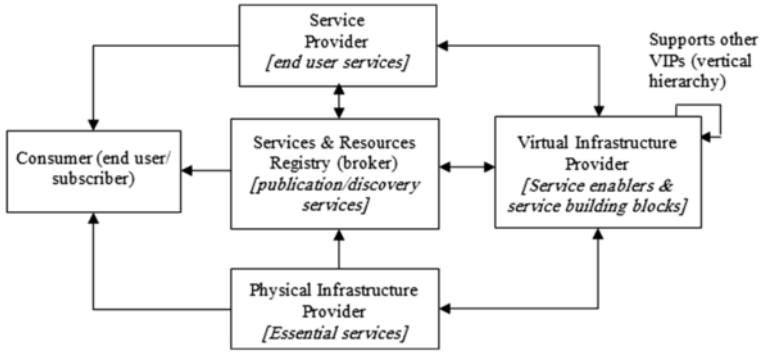

Figure 2 Service-oriented hierarchical business model for virtual networking environment.

control); and *End user services* constituting the value-added services offered to users.

Unlike the current Internet business model that is based on a single role (the Internet Service Provider - ISP), our model distributes the functionalities of the traditional ISP and introduces five business roles, namely: 1) *The Physical Infrastructure Provider (PIP)* that owns and manages a physical network infrastructure and can partition its resources using some virtualization technology. The services offered by the PIP are essential bearer services. 2) *The Service Provider (SP)* that has a business agreement with the subscriber and offers value added services, which could be simple or composite (i.e. formed by combining service building blocks); 3) *The Virtual Infrastructure Provider (VIP)* that finds, negotiates, leases, and aggregates virtual resources from one or more PIPs, deploys any protocols/technologies in the instantiated VN, and operates it as a native network. The VIP supports SPs or other VIPs with service enablers and service building blocks and has no direct business agreement with consumers; 4) *The Consumer* who acts as the subscriber and the end user of value-added services; and 5) *The Services and Resources Registry (SRR)* acting as resource broker by providing information to find other parties and the services/resources they offer.

It is important to mention that while the VIP plays a resource brokerage role to SPs by finding and aggregating virtual resources to form virtual topologies, this role is not to be confused with the information brokerage role played by the SRR. This last focuses on providing information about the existing pool of virtual resources, to facilitate the resources' discovery and selection process.

Integrated hierarchical information model

Figures 3 and 4 show a high level view of our integrated information model, which was presented in [12]. As depicted in Figure 3, our information model revolves around three main concepts and their relationships: *roles*; *services*; and *resources*. Roles are entities that collaborate to offer/consume resources and services and exchange information related to these resources/services. A role can act as resource provider offering and managing virtualized resources, or as a resource consumer accessing virtualized resources. 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). Furthermore, roles are considered to be distributed and loosely coupled entities interacting via programmable interfaces. Finally, just like web services, various levels of network services can be published, dynamically discovered, composed, and used, in our model.

In Figure 4, we model the different roles and their relationships to physical/virtual topologies and various levels of services. 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

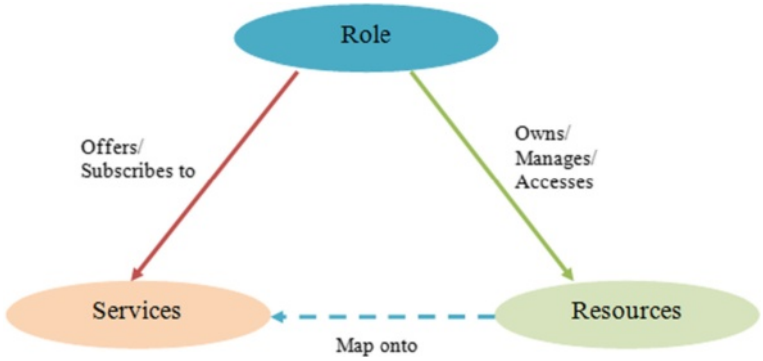

Figure 3 Main concepts of the integrated hierarchical information model.

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

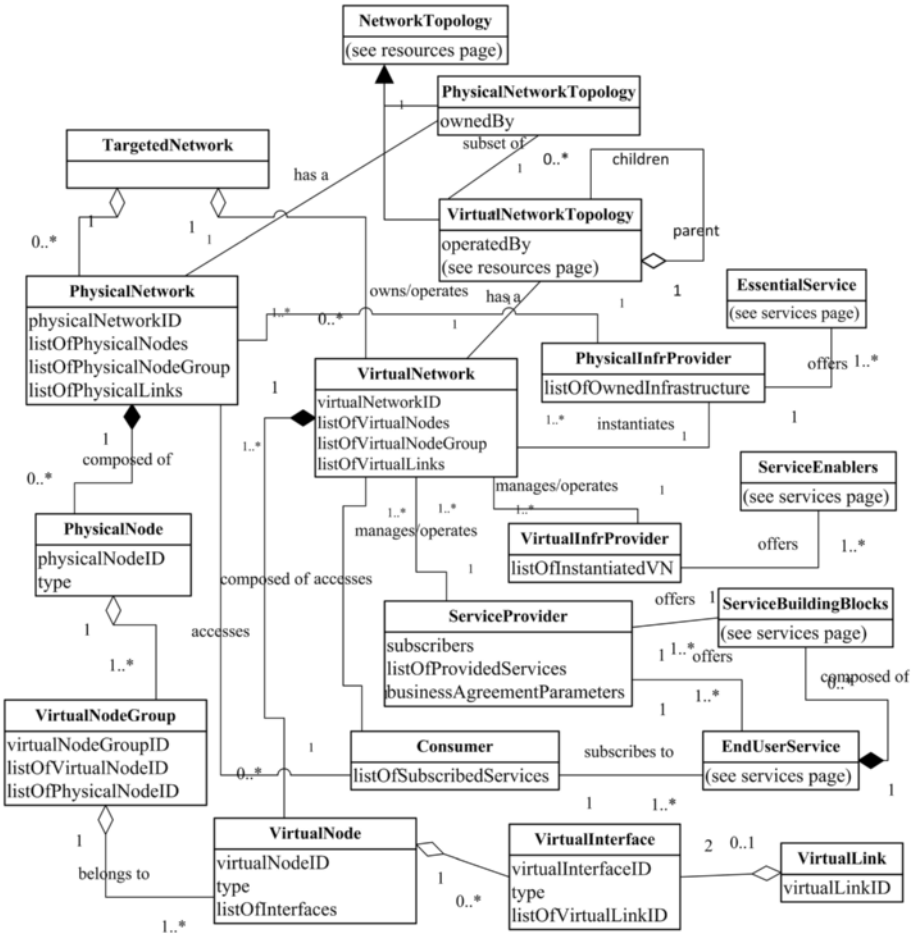

Figure 4 High level view of the integrated hierarchical information model.

To further detail our model, Figures 5 and 6 respectively show the resource-level view and the service-level view.

In the resource-level view, we consider as the basic building component of a virtual network, a *NetworkElement* (NE) 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.). 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 *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. A path 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.

We model *network topology* as *physical*/*virtual* topology. In general, a *network topology* has name, type (i.e. bus, ring), path list, and is composed of a set of *nodes*. 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 hierarchical associations among VNs. Besides, it contains attributes related to availability, start time, period, and a reference to its operator.

In the service-level view shown in Figure 6, 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 SRR; (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 he 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* for example *call control*, *presence*, *conferencing,* and *messaging*; and (4) *Service enablers* defining the support functions needed for the operation of end user services. Examples of service enablers include: *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 service rank, cost, and maximum number of supported users can be expressed as well.

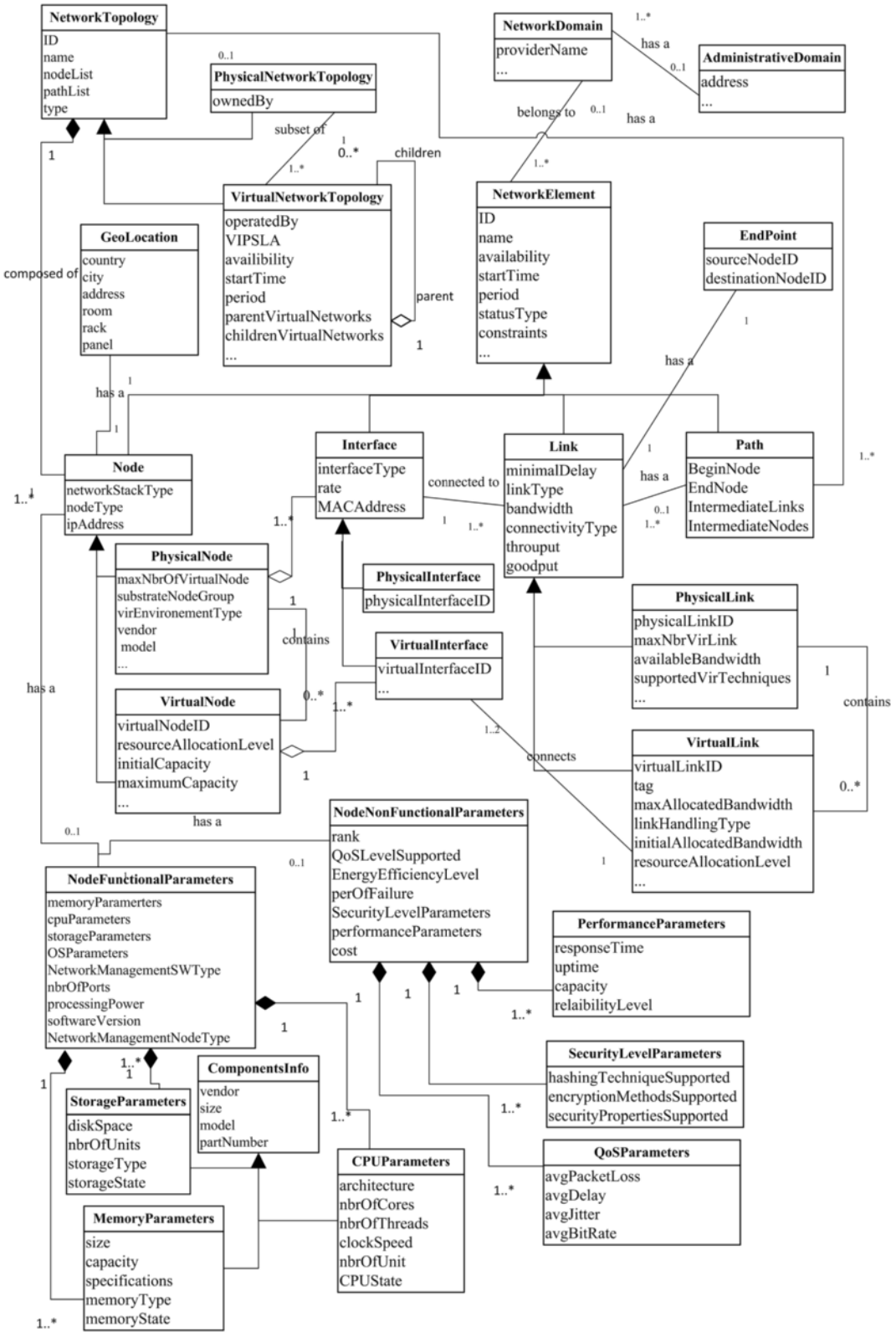

Figure 5 UML-based modeling of physical and virtual networking resources.

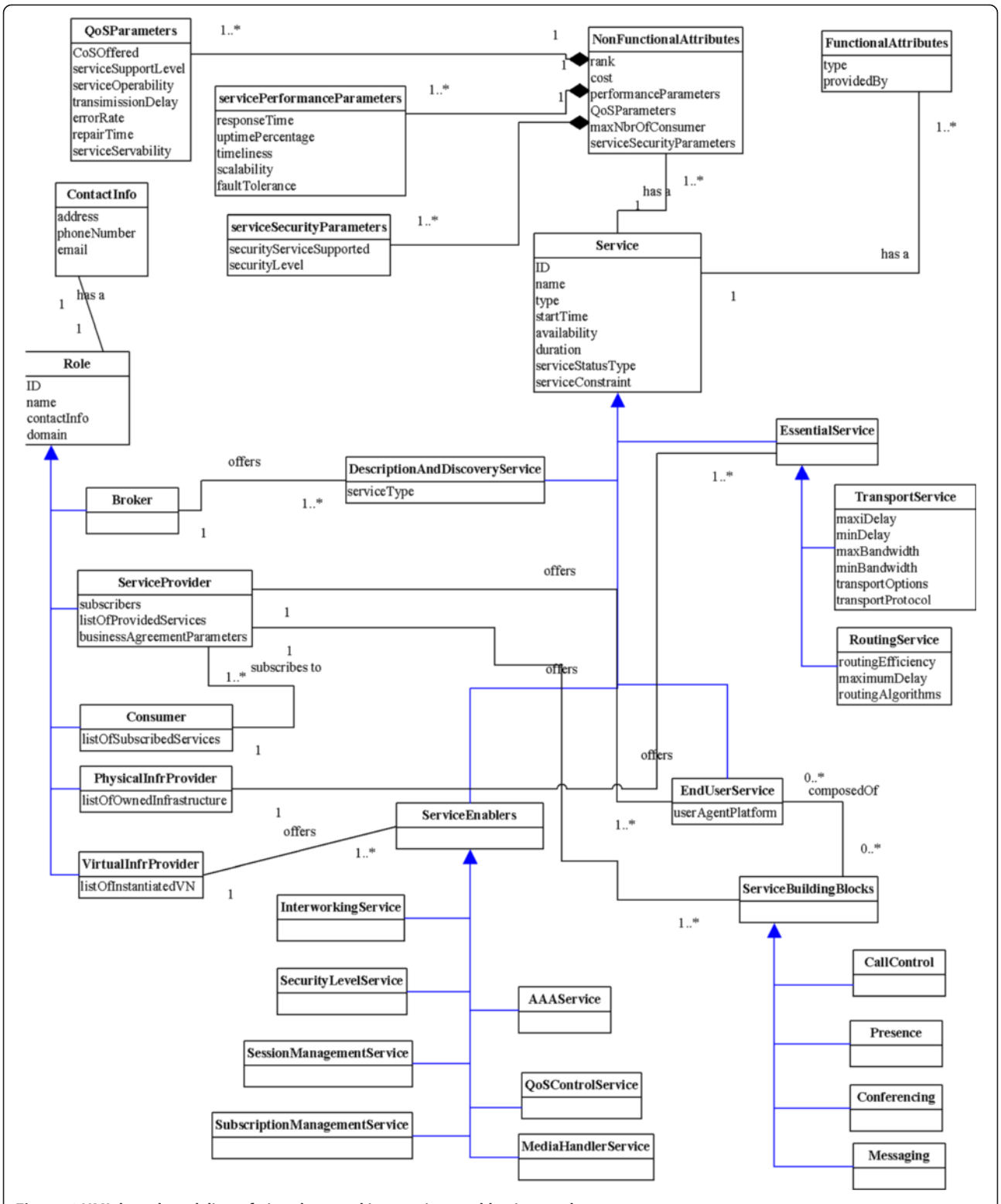

Figure 6 UML-based modeling of virtual networking services and business roles.

Broker-based framework for virtual resource publication and discovery

Figure 7 illustrates the system architecture of the proposed framework.

We selected a broker-based approach aiming at resolving the issue of resources/services' publication and discovery in network virtualization environments. The main objective of our work is to find an efficient solution that enables seamless interaction and collaboration between various roles. Within this framework, Physical Infrastructure Providers are network-related resource suppliers who advertise (i.e. publish) the description of the resources offered into the Broker's service and resource repository. Virtual Infrastructure Providers 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 Providers are end-to-end service providers who require and discover virtual networks on top of 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 sub-sections.

Overall architecture

Figure 8 gives a detailed view 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 the SP at the second virtual level. Consequently, roles depend on each other to perform the virtual network provisioning process. We selected a resource-broker approach to cope with the complexity of managing and organizing resources [22]. We introduce a resource and service broker that serves as mediator while coordinating the communication between various roles.

The resource broker allows roles to publish their resources and discover other roles' resources, in addition

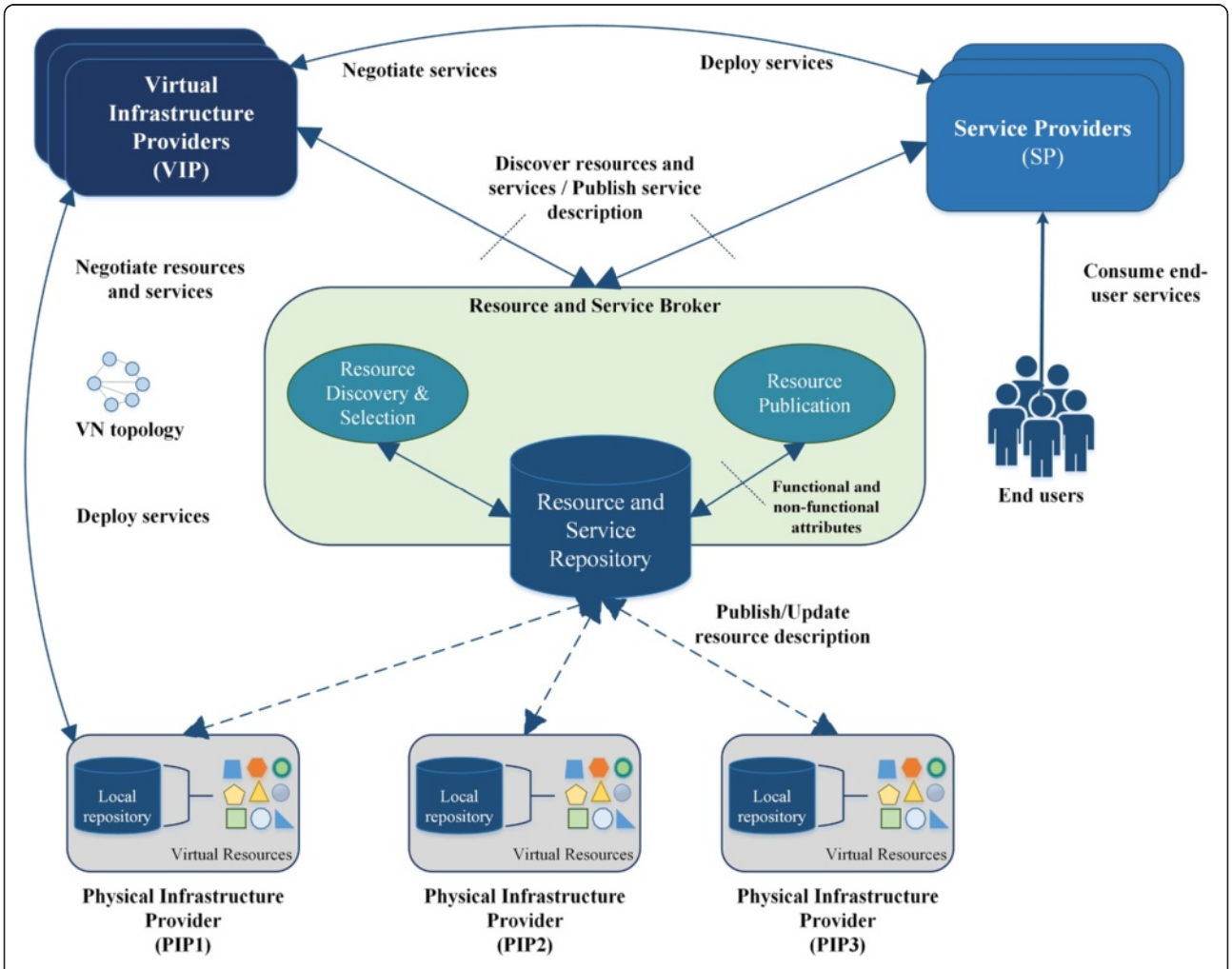

Figure 7 System architecture of the proposed framework for resource publication and discovery.

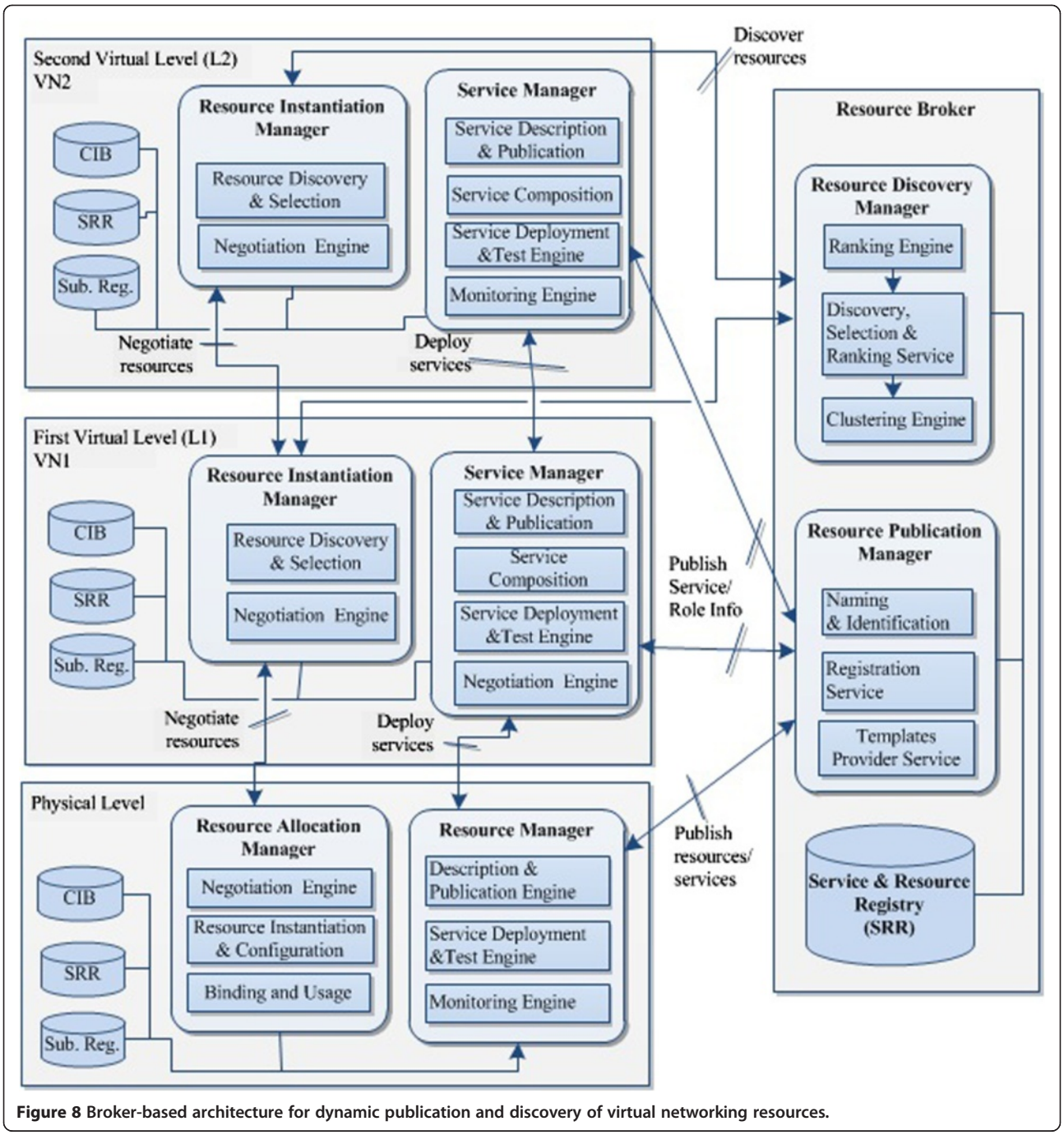

Figure 8 Broker-based architecture for dynamic publication and discovery of virtual networking resources.

to providing the ability to find the most appropriate resource based on particular characteristics and constraints. Thus, it manages the inventory of federated resources in the 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 candidate resources. Requestors, in turn, can perform another selection stage in order to refine the list based on some local preferences (such as QoS, cost). Prior to resource allocation, and to reach an agreement on the selected resources, requestors negotiate resources' related parameters (such as price, availabilities, QoS-related parameters) with resource providers. 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 is published into the resource broker. In this architecture, communication between layers is bidirectional and can be performed through standardized public interfaces (e.g. web services).

Components' description As part of the resource broker's design, we distinguish two key services involved in the resource publication and discovery process: (a) *The Registration Service* enabling the registration, updating, and deletion of information about resources; (b) *The Discovery Selection and Ranking Service* receiving as input a request containing the description/requirements of resources of interest along with 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.

Supporting the resource publication/discovery process, we find four support modules. 1) 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.). 2) To facilitate resource selection, the *Clustering Engine* arranges information about resources contained in the SRR into clusters (grouping resources having similarities). 3) 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. 4) *The Templates Service* provides the different roles with an up-to-date template for describing resources or network services.

Communicating with the broker, the first layer of the hierarchy (L1) provides components for describing, publishing, and instantiating virtualized networking resources as well as negotiating resources with other roles. The second and the top-level layer's components are responsible for describing, deploying, and publishing network services. L1 contains components grouped into the following sub-systems:

The Resource Manager (RM) handles the management and publication of resources and encompasses three components: *1) 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 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. 2) *The Service deployment and Test Engine* enables the PIP to deploy and test essential services such as routing or transport services. 3) The *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 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: 1) *The Resource Negotiation module* handles and coordinates the resource negotiation process with a given virtual layer. *2) The Resource Instantiation and Configuration module* is responsible for the "slicing" of physical resources. It handles the instantiation request and enables the creation and configuration of virtual resources. *3) The Binding and Usage module* maps a virtual resource to a physical one (i.e. maps resources to requests), reserves the allocated resources, and triggers the monitoring process for dynamic resource management purposes.

Since in a NVE, multiple virtual layers can be built on top of physical one, we designed similar components to be used at each virtual layer. However, the type of resource/services being offered is different. At the first and second virtual layers, the *Service Description and Publication Engine (SDPE)* 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* module interacts with the broker on a request-response basis, and performs another stage of resource selection involving some local criteria/constraints. The *Service Composition* module enables the combination of two or more services into a composite service. The *Service Deployment and Test* module coordinates the steps involved in service deployment and performs some tests to validate the virtual-to-physical mapping. The *Service Monitoring* module monitors the status of deployed services to ensure QoS. Finally, the *Negotiation Engine* module conducts the negotiation of resources with one or more PIPs.

Interfaces In an open virtualization environment, the communication between virtualization layers should be conducted through public and flexible interfaces. For our architecture, we have selected REST [23] as a lightweight communication middleware. Our choice is motivated by many reasons. Nowadays, virtualization solutions (e.g. Xen, KVM, VMWare) are provided with RESTful APIs to allow full programmatic control over virtualized resources. In addition, RESTful web services rely on an existing well known standard (HTTP) and allow the

representation of resources in a variety of formats (e.g. XML, JSON, etc.). Furthermore, as opposed to SOAP-based Web Services, RESTful Web Services are simple, lightweight, and easy to develop.

In terms of design concept, REST is not an architecture but rather an architectural style that revolves around the notion of resources and follows the client–server architecture. As per REST's design principles, a resource is anything that can be exposed to the world (i.e. the Web) and be made available/accessible through a uniform interface (URI) just like a web page. Therefore, REST Services are URI-based, and each resource is addressed through a URI but could have many data representations (XML, JSON, binary, etc.) - which facilitate information sharing and interoperability/integration with existing systems. Using the same URI, but different HTTP methods (i.e. POST, GET, and UPDATE), resources can be created, read, updated, and deleted in "CRUD" style. As opposed to Big Web services, RESTful services are not published in a service registry (UDDI) to be later discovered. They are rather made available at uniform paths (or root URIs) that are handed to the requester beforehand. In most cases, services' URIs are provided with the API documentation. A well-defined URI template should be used to identity entities and illustrate their relationships.

In our architecture, the SRR is an information store that holds information about resources, which are arranged in a directory-like structure from a logical point of view. We name the base URL of the services after the following pattern: http://{hostname}/api/{apiVersion}/. Furthermore, we identify the entities on which the services operate using the following URIs: /resources, /services, /networks, /roles, /requests, /offers. Binding one of these URIs to the base URL leads to a service' path, e.g. a "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. In addition, it 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).

Table 1 summarizes the uniform interfaces that are used to create, control, and manage resources. The resources being managed are listed in the first column, while the second column lists their URIs. We find in the last column the HTTP methods applied on the corresponding URI.

Illustrative scenario Figure 9 illustrates the usage of our proposed information model and architecture for dynamic resource discovery and selection, in a secure content distribution scenario.

In this scenario, we find the following roles: a PIP managing the infrastructure offering communication capabilities; a VIP instantiating VN1 to offer security, QoS, and content-based routing as service enablers; and a SP instantiating VN2 to offer the secure content distribution value-added service to consumers. In our scenario, the interactions between the different entities are REST-based, and the following interactions are depicted: resource/service publication, discovery, selection, negotiation, and virtual networks' instantiation. It should be noted that service invocation and usage phase, which includes the security and the content distribution related interactions are out of the scope of this paper, which is focusing on the virtual network instantiation process rather than the end-user service invocation and usage process.

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 and sends back a confirmation message (200 OK) as well as the newly created resources' URIs to RPE (steps 1& 2). To deploy service enablers, a VIP needs to instantiate a virtual network (i.e. VN1) on top of aggregated resources (possibly from different providers). Therefore, the Virtual Resource Discovery and Selection (VRDS) module initiates a discovery request containing the description of the desired resources along with the requested 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) module 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 and publishes the newly created virtual network description in the broker. The RIC sends an acknowledgement 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) module (steps 11 to 13). Upon successfully instantiating the virtual topology

Table 1 Broker web services' APIs

Resources	URI	HTTP action/Description
	Base URL: http://www.ResourceBroker.com/apiV1/	
Management of resources: publication, update and deletion.	/resources	POST: create a new resource in the SRR. GET: get all resources.
	/resources/{resource_id}	GET: retrieve a resource. PUT: update a resource. DELETE: delete an individual resource.
Network services	/services	POST: create a new service. GET: get list of all services.
	/services/{service_id}	GET: retrieve a service. PUT: update a service. DELETE: delete an individual service.
Role information	/roles	POST: create a new service. GET: get list of all services.
	/roles/{role_id}	GET: retrieve a service. PUT: update a service. DELETE: delete an individual service.
Topology (Virtual Network)	/networks/	POST: create a new network. GET: get list of all networks.
	/networks/{network_id}	GET: retrieve a network. PUT: update a service. DELETE: delete an individual network.
Requests (negotiation and service deployment requests)	/requests	POST: create a request GET: get list of requests.
	/ requests/{request_id}	GET: read a request. PUT: update a request. DELETE: delete an individual request.
Offers (negotiation offers)	/offers	POST: create a new negotiation offer. GET: get all existing offers.
	/offers/{offer_id}/	GET: read an offer. PUT: update the information of an offer. DELETE: delete an individual offer.

(resulting in the creation of VN1), 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 VN2, and the deployment of the content distribution end user service offered by the SP are somewhat similar to the steps performed to instantiate VN1. 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).

Proof-of-concept prototype

Prototype architecture

Figure 10 depicts the software architecture of the implemented prototype and the technologies. Only a subset of the components proposed in section 3.3.1.1 was implemented. For simplicity reasons, we combine the VIP and SP roles.

Our implementation consists of three management nodes, namely: the *PIP Management Node (PMN)*; the *VIP Management Node (VMN)*; and the *Broker Node (BN)*. Each node holds a repository that contains resource information and hosts the application logic that realizes the functionalities of the corresponding roles (e.g. PIP, VIP, and Broker). This application logic is a set of software modules written in the Java programming language and provides JFC/Swing-based user interfaces for the administrators.

We use XML to describe the resources and formulate the various requests (e.g. discover and negotiation requests), and XSD (XML-Schema Definitions) to define the structure of the data models and specify constraints on the data contained in the XML documents. Each document exchanged between two roles is a data model (an instance of our proposed information model).

We selected Jersey [24], an open source JAX-RS (JSR 311) reference implementation, to implement the REST interfaces, and Grizzly web server [25] to deploy the web services. Moreover, we used JAXB 2 [26] for marshaling

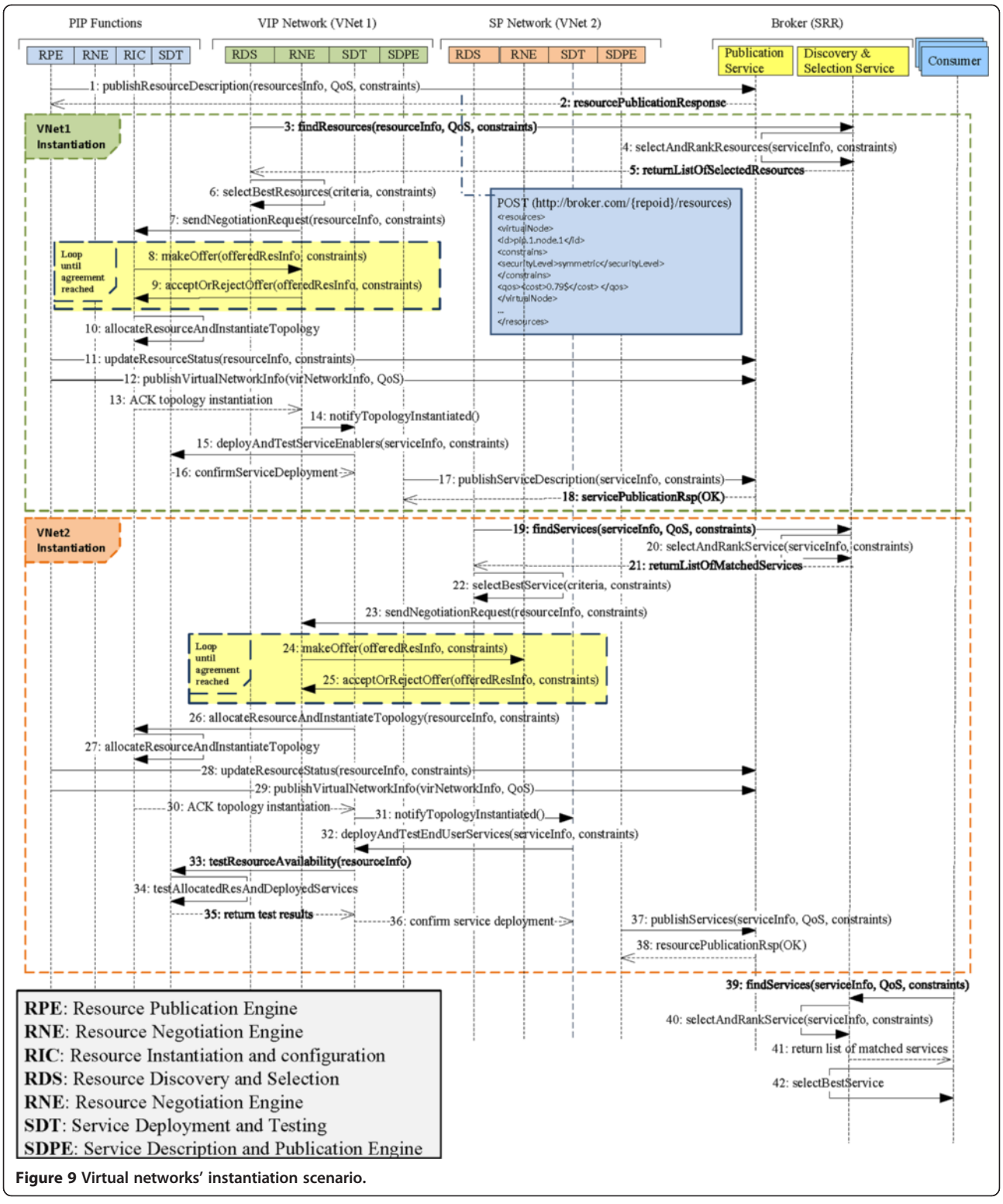

Figure 9 Virtual networks' instantiation scenario.

and un-marshaling the XML data contained in REST messages' body.

In this implementation, the BN is the key node encompassing a resource naming/identification module, as well as ranking and clustering engines, which are involved in the resource publication and discovery processes. In our approach, we store resource properties such as node type (e.g. VM, vRouter), operating system type, and

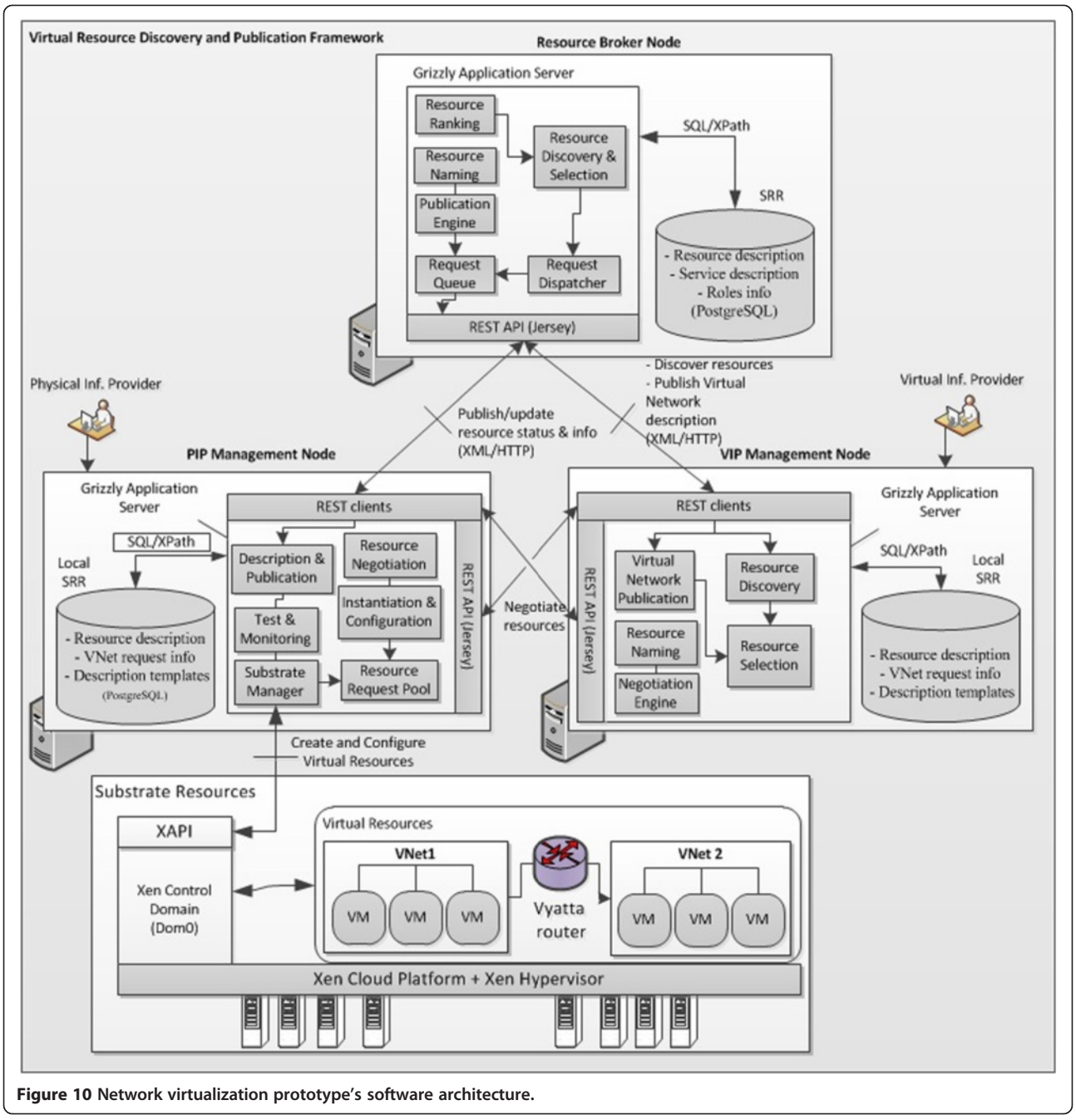

Figure 10 Network virtualization prototype's software architecture.

virtualization environment, in separate columns in the database. The document containing the resource description is stored in XML format in the same database. When received, resource publication and discovery requests are first stored in the Request Queue and later forwarded by the Request Dispatcher to the appropriate module. We use a 32 digits-based identification scheme to identify each advertised resource. The issued identifier consists of three parts: the first set of three digits identifies the provider (PIP, VIP or SP), the second set of 13 digits determine information such as the resource type, the virtualization environment being used and the remaining set of 16 digits are GUID-based (automatically generated).

In this work, the discovery request contains two parts: the first part is selection parameters such as OS type, node type, virtualization environment and the second part is a set of selection constraints that could be applied on functional attributes such as CPU and memory. To select the optimal resources, the Resource Discovery and Selection Engine (RDS) queries the repository to get a set of resources having similarities in their description.

In such a query, the selection parameters described in the discovery request are taken into consideration, which helps in filtering the resources that do not match part of the request. Afterwards, the RDS processes the returned set of resources to evaluate their functional attributes if they correspond to the selection constraints specified in the discovery request. For instance, a constraint on the desired CPU's processing speed could be defined as a range between two values (minimum and maximum).

The PMN sends resource publication requests to the BN, and processes virtual network instantiation and resource negotiation requests for the PIP. It uses a local database to store and manage resource information and description templates. Furthermore, the PMN monitors allocated resources and updates their status information in the broker. In addition to other components, the PMN architecture includes a Resource Instantiation and Configuration engine that handles virtual resource instantiation, configuration, and testing. This engine allows for managing and controlling the substrate resources (detailed later in this section).

Finally, in addition to discovering the resources needed to deploy end-user services, the VMN interacts with the PMN to negotiate resources. To build the locals and the broker databases, we first selected eXist-db [27] - a native XML database. However, we conducted some performance and scalability experiments to evaluate eXist-db's ability of processing and storing a large number of resources. Those experiments demonstrated that a native XML database is not suitable for our prototype due to the lack of flexibility in using the tools exposed to store and retrieve information in it. Consequently, we selected the open source RDBMS PostgreSQL [28] that offers native XML support for storing XML documents, SQL/XML publishing/querying functions, full-text search, as well as full-text indexing and XPath support. Furthermore, PostgreSQL stores an XML document in its text representation, which results in fast information retrieval and adds flexibility in terms of resources' description by eliminating the need to change tables' schema whenever additional information is added to the document. Upon receiving a resource publication request, the publication engine validates and parses the resource description, and stores the received document in its XML text format in the database. Resources are indexed based on their identifier that is stored in a separate column. This enhances the selection process by eliminating unnecessary parsing of an XML document, since the resource identifier contains information about the type of resource. We used Xen Cloud Platform (XCP) [29] that includes the Xen Hypervisor as well as Xen API (Xen Management API or XAPI) for virtualizing substrate nodes. Based on para-virtualization principles, Xen has demonstrated to be the virtualization platform of choice due to its capabilities in terms of performance, features, and isolation level among virtual machines. XAPI provides programmatic access to, and remote administration of, Xen-enabled virtual resources through XML-RPC services.

We implemented the Substrate Manager (SM) using XenServer's SDK that is provided by Citrix. The SM is responsible for automatically instantiating a virtual topology as described in the VN request. We automated the resource provisioning process by eliminating the human intervention needed to create the requested virtual resources and configure their network settings. For this matter, we prepared a set of virtual machine templates on which we deployed Shell scripts that enable the addition or removal of Ethernet interface(s), changing a VM's IP address, as well as setting/removing a static route between two nodes (in case of a virtual router). In order to execute such scripts, the SM uses an SSH connection to the targeted virtual machine. In addition to creating and configuring virtual resources, the SM monitors the status of the running resources and displays their dynamic attributes on the PIP's interface. In this implementation, we selected Vyatta [30] virtual router as shown in Figure 10 to connect two or more virtual networks.

Prototype setup and test scenarios

As shown in Figure 11, the experimental setup consisted of two management nodes (one PMN and one VMN), one broker node, and four nodes that represent substrate resources. The PMN and VMN and the substrate nodes are DELL Precision 390 machines equally equipped with Intel Core™ 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™ 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. 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 running the experiments, we generated a set of resource description XML 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.

We successfully tested the interactions related to the virtual network instantiation scenario as depicted in Figure 11.

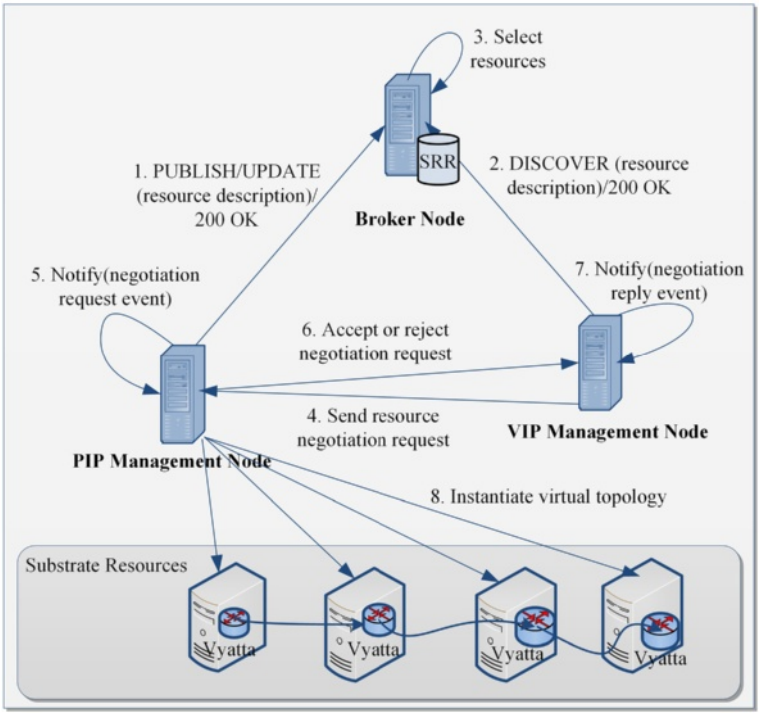

Figure 11 The network virtualization prototype setup.

First, the PMN published the description of the virtual machines and Vyatta routers that we installed on the substrate resources to the broker. Then, the VMN sent a discovery request to the broker. Afterwards, the broker node retrieved the information needed as described in the request from its resource repository and selected the resource candidates. After receiving the selected resources, the VMN triggered a negotiation process by sending a request to the PMN, which when detected, a notification message was displayed on the PIP's console. The negotiation process went through two phases: First, the PMN rejected the offer and sent back the request to the VMN; then the VMN sent another request which was accepted by the PMN. Upon reaching an agreement, the PMN instantiated the virtual topology and started the virtual resources (using XAPI client). When the requested resources started successfully, the PMN updated their published information in the broker. Figures 12, 13 and 14 illustrate three the screen shots of our prototype's operation – namely the VIP resource discovery view, the PIP resource publication view, and the PIP virtual topology management view.

Basic performance evaluation

To assess the basic performance of the prototype, we used the setup described in the previous section and evaluated the interactions related to resource publication (between the PMN and the BN), resource discovery (between the VMN and the BN), resource negotiation (between the VMN and PMN), and resource instantiation (between the PMN and the machines representing substrate resources). We used JMeter [31] to evaluate the REST APIs' performance, and we modified the application logic that is deployed on the management nodes to add support for measuring internal operations' processing times.

Table 2 shows the evaluation results. Each result represents the mean value calculated over 40 trials, in addition to some statistical distribution related results (such as the standard deviation and the confidence level) giving indications about the variability of the mean values presented.

In the table, the response time for resource publication is calculated at the PMN as the difference between the time when the PMN's publication module sends a publication request and the time it receives a response from the BN. The time for publishing a resource includes the time taken to extract description of resources from the REST message's body and the time to store it in the broker's repository. The results shown in the table are the average measurements over 40 trials. For each trial, we sent one resource publication request containing a document describing 2 virtual resources. On average, it took 204.25 ms to process this publication request, which generated 25.89 Kbytes of network load – values that we consider as reasonable. However, as we increased the number of publication requests, the response time and network load measurements increased. This is due to the request processing overhead and the concurrent access to the

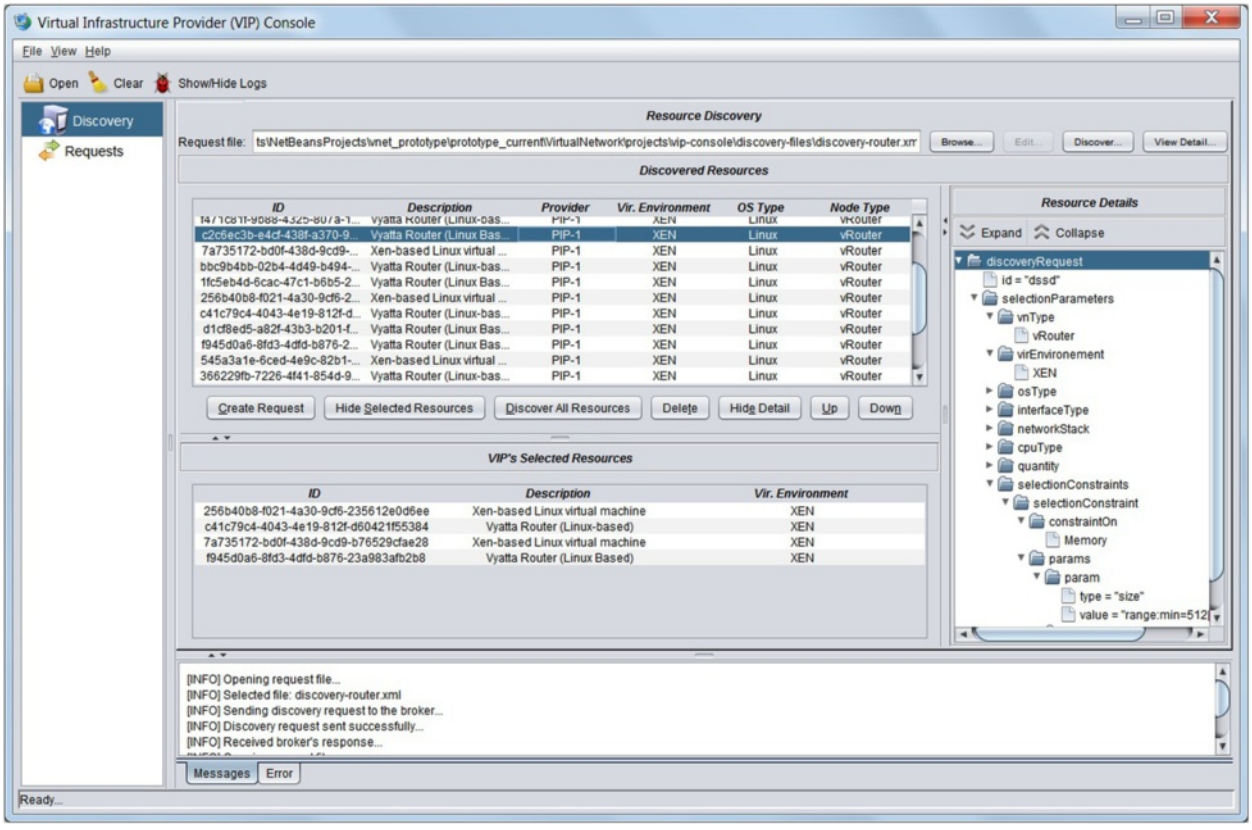

Figure 12 Prototype operation - VIP resource discovery screenshot.

resources' database. The load and stress testing results will be presented in the next sub-section.

The resource discovery response time, which gives an indication about the performance of the selection algorithm, is calculated from the moment the VMN's discovery module sends a discovery request until it receives a response with the selected resources. This includes the time used for the execution of the selection algorithm and the database query time to get the list of potential resources. To perform the resource discovery experiments, we populated the resources' repository with the descriptions of 5000 different resources. The results shown in the table are related to the tests done with one resource discovery request of two virtual resources and 50 processed resources during the selection process. On average, it took 181.1 ms and 23.61 KB of generated network load to process such a request. Additional tests show that as the number of discovered resources increases, the response time and the network load increase as well, due to the increased number of resources that are taken into account by the selection algorithm and the increase in size of the list of matched resources that is sent back.

The resource negotiation response time, measured at the VMN level, is calculated from the moment the VMN's negotiation module sends a negotiation request until the response is received from the PIP. On average, it took 186.4 ms and 34.16 KB of generated load to process a resource negotiation request related to two virtual resources.

Finally, for virtual topology instantiation, the response time is measured at the PMN level from the moment a VNet instantiation request is received until the booting of the virtual machines and the configuration of their virtual interfaces (through the XAPI client) is completed. In our test scenario, the virtual topology consisted of four Vyatta virtual routers connected by three links, as shown in Figure 13. On average, it takes one minute and 10 seconds to create and configure a Vyatta virtual machine, while it takes 5 minutes 58 seconds to create and configure a virtual topology consisting of four Vyatta virtual routers and three virtual links.

Analyzing those results, we conclude that the system yields an acceptable performance for the recurring operations (i.e. resource publication, discovery, and negotiation) – The response time for those operations ranging from 181 ms to 204 ms, while the generated network load ranged between 24 Kbytes and 34 Kbytes. As for the virtual topology instantiation operation, it does result in a significant response time due to its nature that requires the creation and configuration of virtual machines and their connection to form the requested topology.

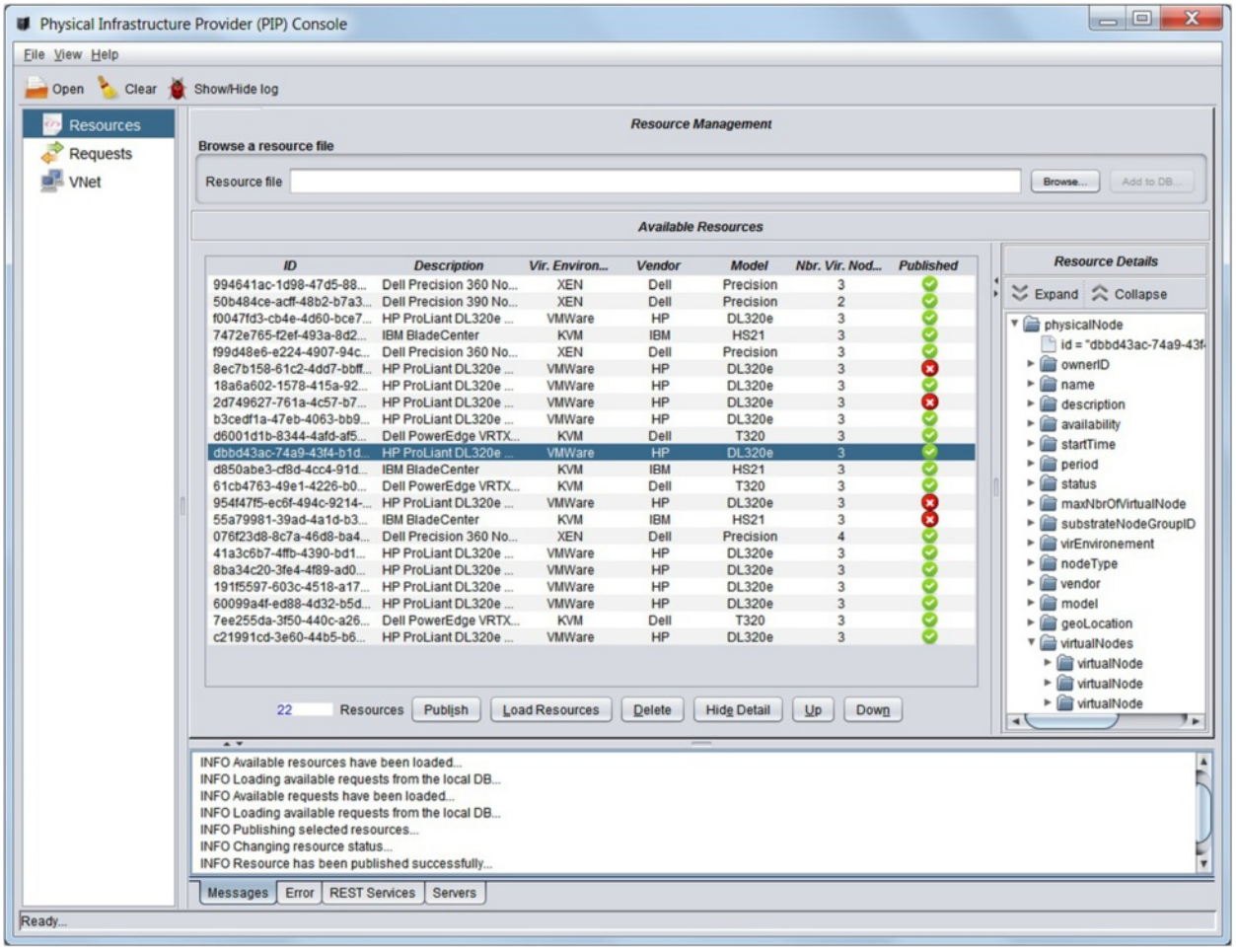

Figure 13 Prototype operation - PIP resource publication screenshot.

However, this operation is only required once, when the VNet is created. Furthermore, the automation of virtual resources configuration using SSH and shell scripts eases and speeds up the virtual topology instantiation process.

Load testing

In order to evaluate the behavior of the system under variable loading conditions, we conducted some load tests using the test setup shown in Figure 11. Figures 15, 16 and 17 show the load testing results for the resource publication, discovery, and negotiation operations.

The resources' publication operation involves the following steps: 1) at the PIP side – loading of the resources description document from the local DB, its validation, and the creation of a REST PUT message containing the description document and its sending to the broker; and 2) at the Broker side – extraction of the resources document from the received PUT message, it's processing and validation, its storage in the DB, and the sending of an HTTP 201 response message to the PIP. As shown in Figure 15, the resources' publication operation shows a polynomial (quadratic) growth pattern in terms of response time, which ranged from 204 ms for 1 publication request to 1 minute and 50 seconds for a 1000 publication requests. This polynomial response time growth pattern can be attributed to three time consuming steps related to resources' publication, namely: the concurrent access to the broker's DB for storage of different resource description documents; the publication messages' processing; and the marshaling and un-marshaling of XML documents. As for the generated network load, it showed a logarithmic growth pattern with values ranging from 26 KB for 1 publication request to 240 KB for a 1000 publication requests. The network load's slow growth pattern can be explained by the fact that the publication requests generated in this test all carried a small XML payload (description document of 2 virtual resources), thus not imposing a high overhead on the network.

As for the resources' discovery and selection operation, it involves the following steps: 1) at the VIP's side - loading and validation of the XML documents containing the description of the resources requested, as well as creation

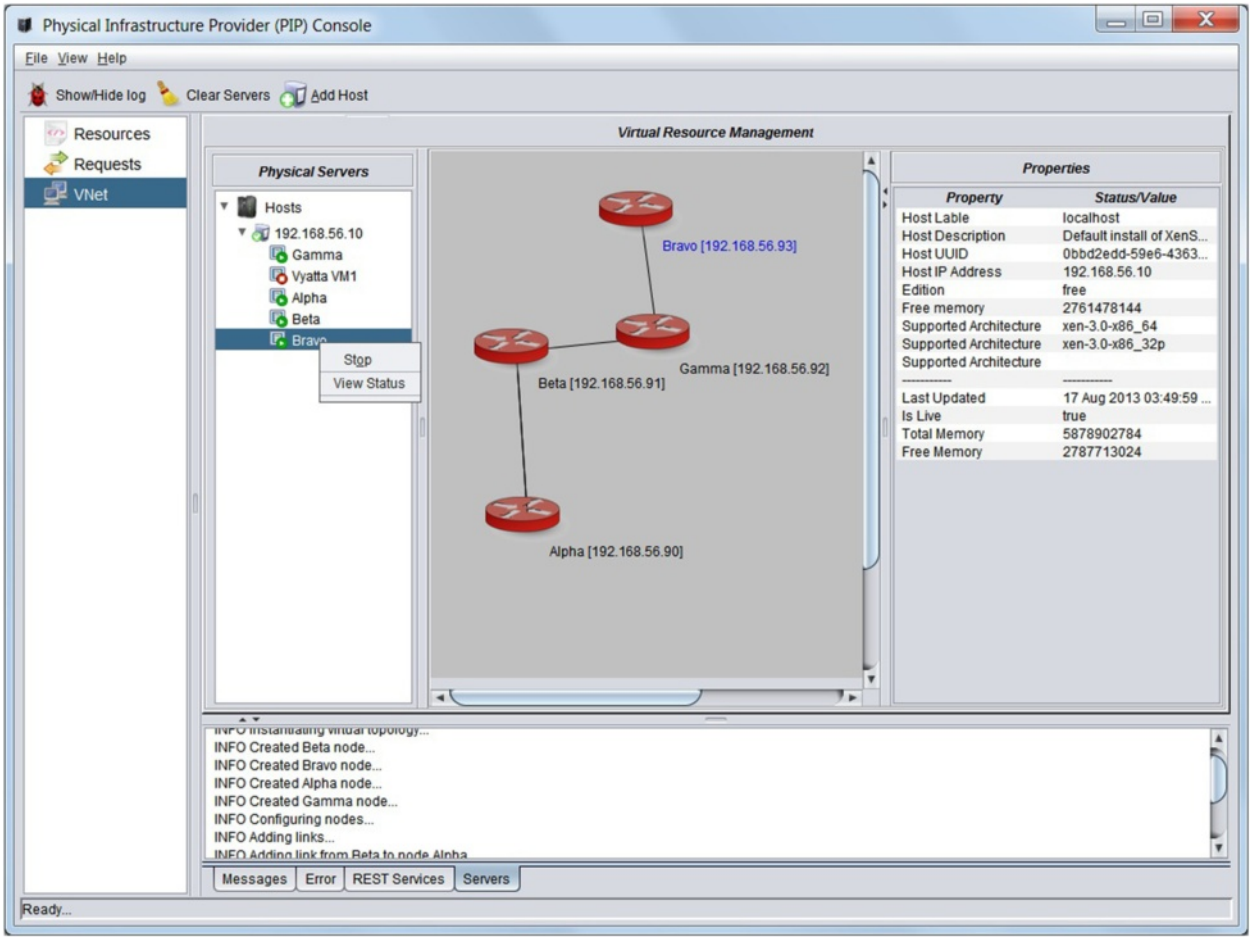

Figure 14 Prototype operation - PIP virtual topology management screenshot.

and sending of a GET REST request with the loaded resource discovery document to the broker; and 2) at the Broker's side - extraction of the resource discovery document from the received discovery request and its processing/validation, execution of the resource selection algorithm to select the resources that comply with the request, and sending of the list of matched resources to the VIP Node. As shown in Figure 16, the resources' discovery and selection operation shows a polynomial (quadratic) growth pattern in terms of both response time and generated network load. The quadratic trend line for the response time gives insights about the performance of the broker's selection algorithm. As shown in the figure, the response time shows a faster increase as the number of resources processed during selection increases – with values ranging from 181 ms for 2 discovered resources/50 processed resources during selection, to 17 seconds for a 1000 discovered resources/5000 processed resources during selection. As for the network load's quadratic trend line, it is associated with the number of resources discovered, with values ranging from 23 KB (for 2 discovered resources) to 1.37 MB (for a 1000 discovered resources). As the number of discovered resources increases, so does the size of the XML payload carried by the response message sent back by the broker.

Finally, the resources' negotiation operation consists in the following steps: 1) at the VIP side – generation of resource negotiation document, creation and sending of REST PUT request containing the negotiation document, to the PIP node; and 2) at the PIP side - extraction/processing of the negotiation document embedded in the received negotiation request, checking the availability of resources, changing the status of the request to *processed* and marking the negotiation document as *accepted*, and embedding the accepted negotiation document in a REST PUT message that is sent back to the VIP node. As shown in Figure 17, the response time for the negotiation operation follows a quadratic growth pattern, while the network load follows a logarithmic growth pattern. The response time's quadratic growth pattern can be explained by the delays caused by concurrent access to the PIP's local repository for updating the

Table 2 Network load and response time measurements

Operations	Interactions	Response Time – Mean value (ms)	Response Time – Statistics		Network Load – Mean value (KB)	Network Load – Statistics	
Resource Publication [1 request/2 virtual resources published]	*PMN – BN*	204.25	Standard Deviation	11.39656	25.89	Standard Deviation	1.371093
			Confidence Level (95.0%)	5.333754		Confidence Level (95.0%)	0.641691
			Minimum	189		Minimum	23.6
			Maximum	231		Maximum	28.3
Resource Discovery [1 request/2 virtual resources discovered/ 50 resources processed during selection]	*VMN – BN*	181.1	Standard Deviation	11.0305795	23.61	Standard Deviation	0.9623983
			Confidence Level (95.0%)	5.162470119		Confidence Level (95.0%)	0.4504162
			Minimum	156		Minimum	22.1
			Maximum	197		Maximum	25.3
Resource negotiation [1 request/2 virtual resources negotiated]	*VMN – PMN*	186.4	Standard Deviation	6.23572053	34.16	Standard Deviation	0.9275888
			Confidence Level (95.0%)	2.91840704		Confidence Level (95.0%)	0.4341249
			Minimum	174		Minimum	32.4
			Maximum	198		Maximum	35.8
Virtual Topology instantiation [4 virtual routers, 3 virtual links]	*PMN – substrate nodes*	358445.9	Standard Deviation	3956.246	143.465	Standard Deviation	0.9051170
			Confidence Level (95.0%)	1851.58		Confidence Level (95.0%)	0.4236078
			Minimum	351670		Minimum	141.8
			Maximum	367280		Maximum	145.3

negotiated resources' status and the negotiation messages' processing. As for the network load's slow logarithmic growth pattern, it can be explained by the small payload carried in resource negotiation messages.

Stress testing

In order to evaluate the behavior of the system under heavy load conditions, we conducted some stress tests, focusing on the publication and discovery related interactions. As test setup, we built a LAN consisting of 5 machines connected by a Cisco Catalyst 2950 series switch. One of those machines (HP Z210 workstation) acted as the Broker, while the other four machines (DELL 390) acted as either a PMN or a VMN (depending on the test scenario). Different test scenarios in which the nodes' roles and the number of generated requests were varied were conducted. Figures 18 and 19 show the stress testing results for the resource publication and discovery operations.

Analyzing the stress testing results, we notice that Grizzly is a suitable application server for the hosting of the broker node, due to its robustness and ability to handle a very large number of simultaneous requests (up to 2000 requests/sec can be supported). Due to those capability, our broker was able to handle very high traffic loads, without crashing. In fact, the system was tested for up to 15,000 publication requests (describing up to 120,000 resources) without failure. As the number of publication request increased, the response time to process the requests increased in a quadratic fashion, while the network load increased in a logarithmic fashion, when two PIP nodes were used as message generators. However, this pattern changed to a cubic growth pattern (for both network load and response time) when four PIPs were used to generate publication requests simultaneously, thus doubling the number of requests generated and the number of resources published. This polynomial (cubic) increase in response time and network load is due to several factors such as database overhead caused by reading/writing records, resource description marshaling and un-marshaling, HTTP requests processing overhead, and increase in the number of requests exchanged.

As for the discovery operation, the broker was successfully tested for up to 12,000 discovered resources (as shown in Figure 19), and the response time and network load both showed polynomial (quadratic) growth patterns with respect to the number of discovered resources, for both Two nodes and Four nodes setups. For 12,000 resources, the response time reached 23.7 minutes, and the generated network load reached 44.2 MB, due to the resource property information that is embedded in the response message.

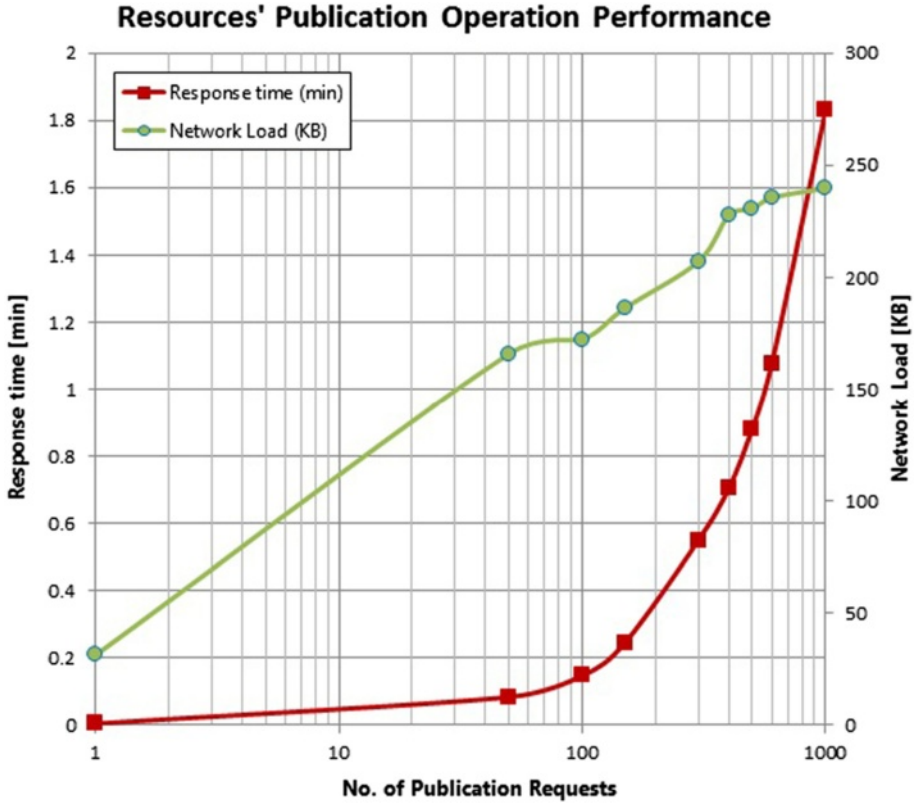

Figure 15 Load testing results for resources' publication operation.

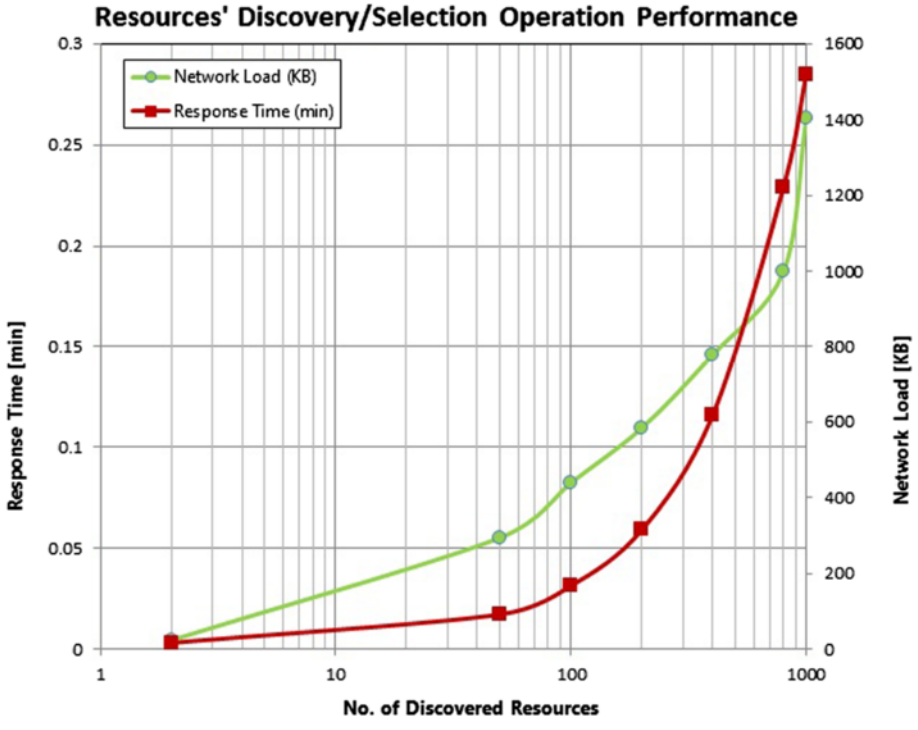

Figure 16 Load testing results for resources' discovery operation.

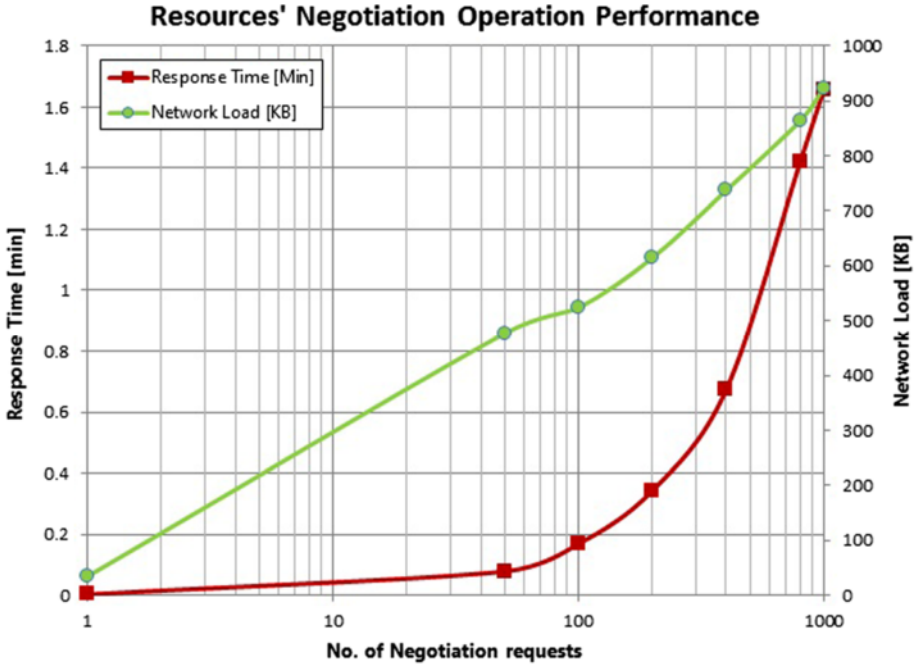

Figure 17 Load testing results for resources' negotiation operation.

Comparative performance analysis

As discussed in Section 2.3, existing virtual resource discovery approaches [7-10] do not rely on an intermediary resources' broker role, and follow a distributed approach in which a VIP must gather information directly from multiple PIPs, before initiating the resources' selection process. In order to compare the performance of our centralized, broker-based architecture with the performance of the distributed, broker-less architectures proposed in the literature, we modified our prototype to operate in a decentralized fashion. This was achieved by removing the broker node and modifying the PMN behavior so that the PIP publishes it resources' description information to a local repository/broker (instead of the centralized BN). As for the VMN, it was modified to enable direct communication between the VIP and the PIP for the discovery of resources' information. Furthermore, the resource selection algorithm, which was originally executed by the centralized BN when it received a resource discovery request, was ported to the VIP node, which now takes care of resource selection following the discovery of candidate resources. Figures 20 and 21 illustrate the centralized and the distributed test bed setups used to collect the comparative performance measurements, while Figures 22 and 23 depict the collected results.

As shown in Figure 22, both the centralized and the distributed architectures achieve similar response times for the resource discovery operation, when the VIP is communicating with one PIP. However, as the number of PIPs increases, the broker-based centralized architecture shows a significant improvement in terms of response time, when compared to the distributed architecture. Indeed, in the centralized architecture, as the number of PIPs increases, the time it takes to discover resources increases in a slow rising linear fashion, while the distributed architecture exhibits a more rapid, quadratic growth curve. While it took 6846.2 ms to discover information related to 10 PIPs in the distributed broker-less architecture, it took 487 ms to discover the same information in the centralized broker-based architecture *(i.e. a 92.8% performance improvement).* This is due to the fact that in the distributed architecture, the VIP had to communicate with the 10 PIPs to gather their resources related information, then perform the selection locally, while in the broker-based architecture, the VIP communicates only with one node (the resources' broker) that categorizes, matches, and selects the most suitable resources (based on (non) functional parameters) to be returned to the VIP node. It should be noted that, in the distributed architecture, prior to the resources' discovery phase, the VIP should discover the contact information of the PIPs (via a public repository) in order to be able to communicate with them.

As for the network load's comparative performance measurements, we observed linear growth patterns for both the centralized and the distributed architectures,

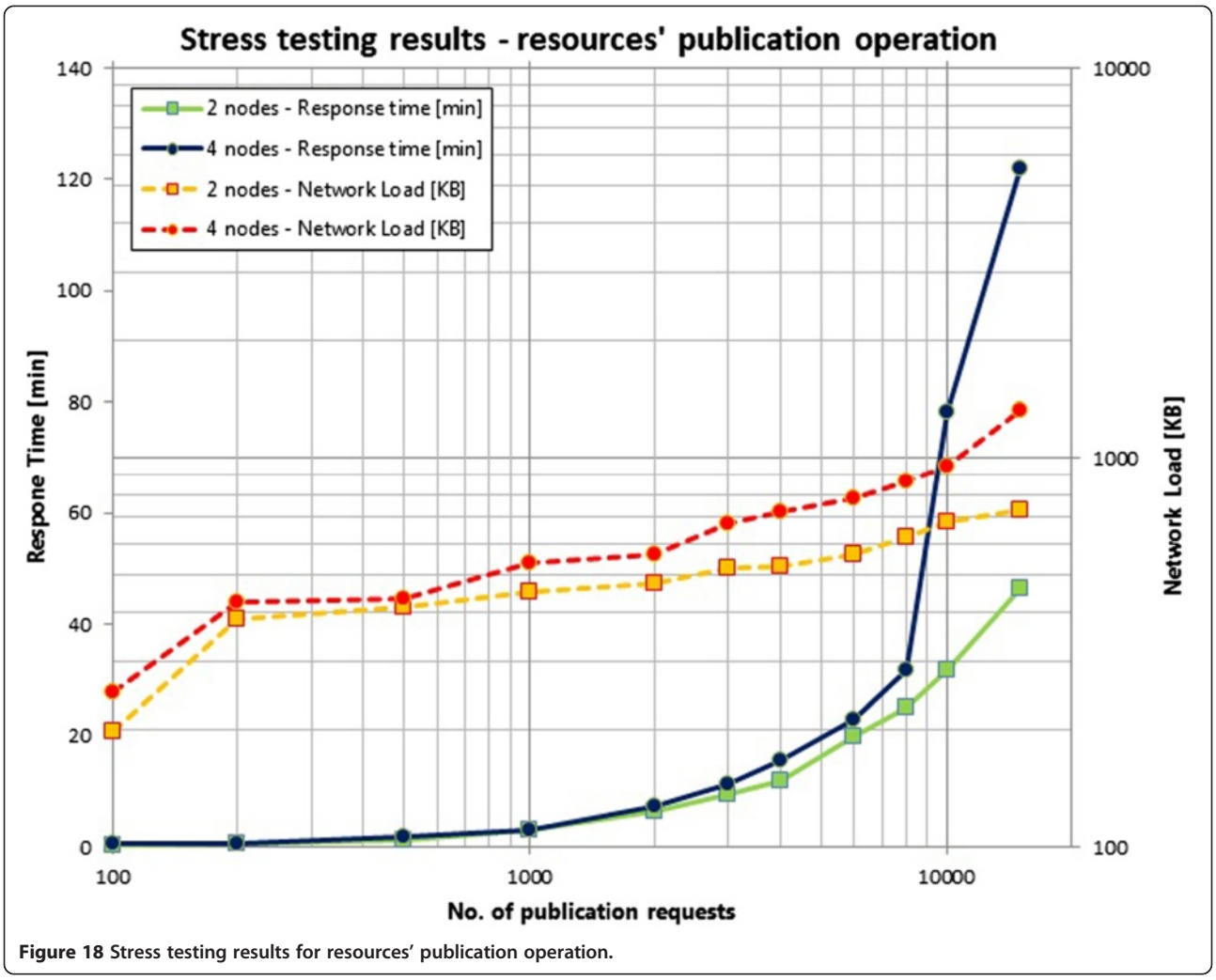

Figure 18 Stress testing results for resources' publication operation.

with a smaller change rate (slope) for the centralized architecture with respect to the distributed one. In fact, for discovery of resources related to 10 PIPs, the network load generated in the distributed architecture was 868.9 KB, vs. 197.4 KB generated in the centralized architecture for the same scenario – *i.e. a performance improvement of 77.3%.* We also noticed that the generated network load for the case of 1 PIP is higher in the distributed architecture, when compared to the centralized one. This is due to the fact that, in the centralized architecture, the intermediary broker node performs an initial selection operation, in order to return the most relevant resources (satisfying functional requirements and constraints on dynamic attributes), which results in a more refined resources list and thus a reduction in the size of the resources' description document returned to the VIP. In the distributed scenario case, since the selection is performed by the VIP, the PIP only performs a simple matching operation (based on functional attributes only), thus returning a less refined and larger list of resources to the VIP.

Based on those results, we can conclude that our proposed broker-based virtual resources discovery architecture offers significant performance improvements, in terms of response time and generated network load, when compared to the existing distributed resources discovery architectures presented in the literature. In fact, in a large scale virtual networking environment in which many PIPs offer virtualized resources for lease, introducing a resources' broker as intermediary role offers benefits in terms of reduced complexity of the resources' discovery operation and the VIP node's logic, as well as improved response time and communication overhead (when a VIP is communicating with a large number of PIP candidates) – thus improving the overall efficiency of the VN embedding process.

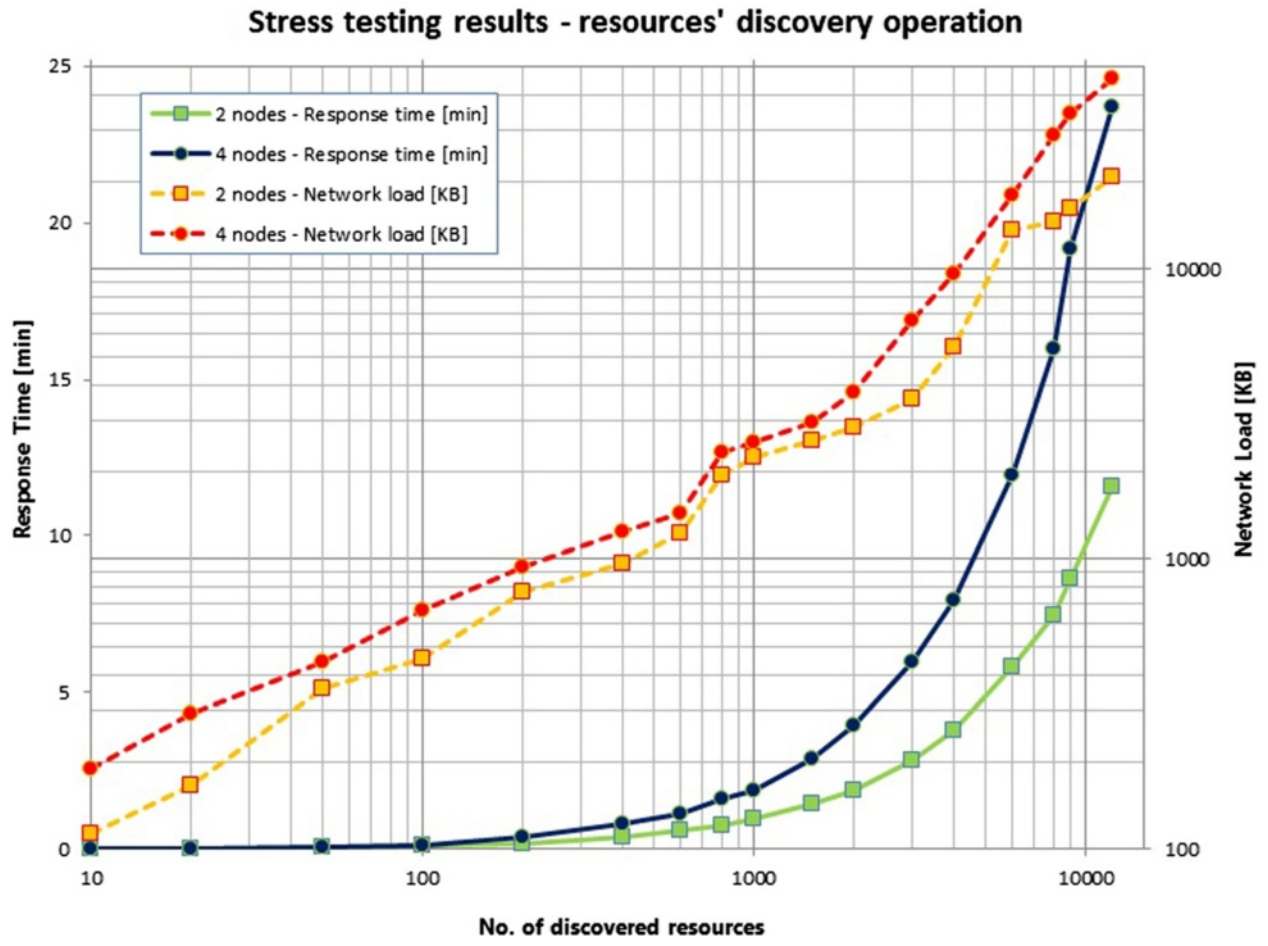

Figure 19 Stress testing results for resources' negotiation operation.

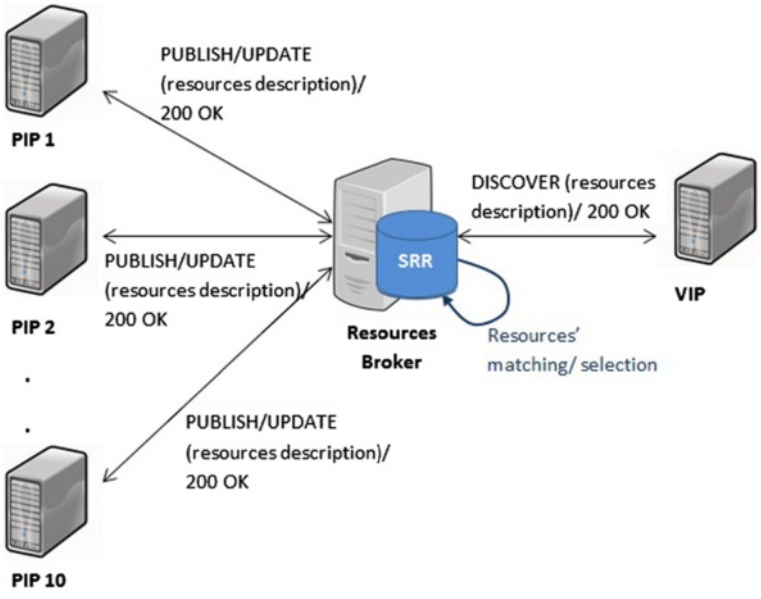

Figure 20 Broker-based architecture test bed.

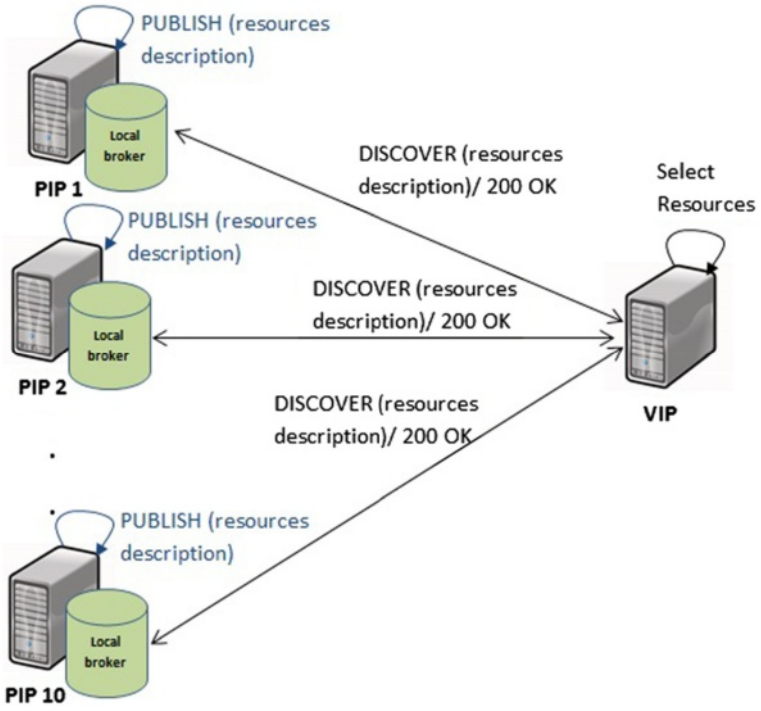

Figure 21 Distributed architecture test bed.

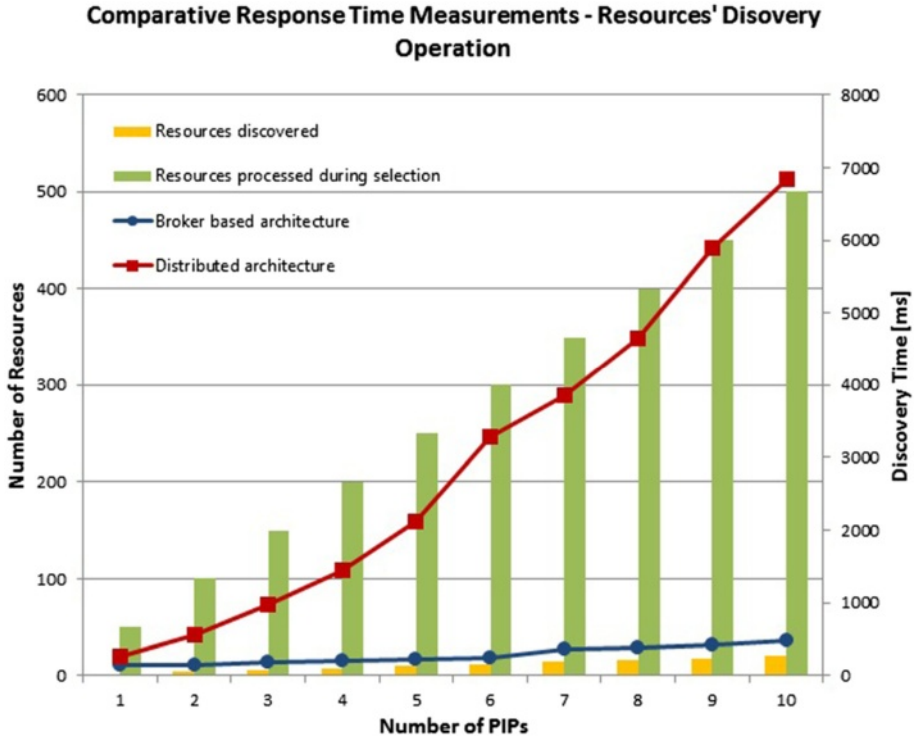

Figure 22 Comparative response time measurements.

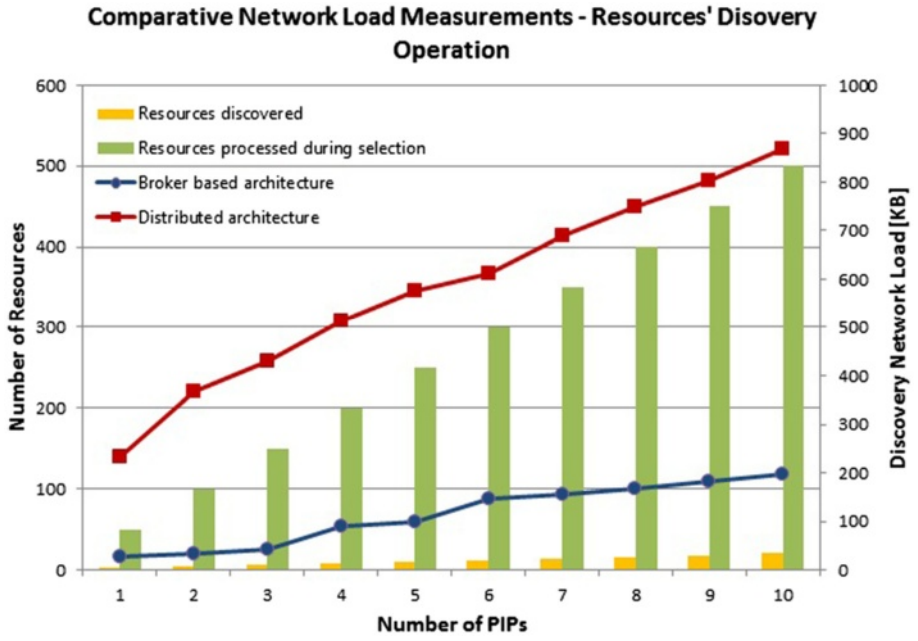

Figure 23 Comparative network load measurements.

Conclusions

Although network virtualization has received considerable attention lately and is seen as a promising way to overcome the limitations and fight the gradual ossification of the current Internet architecture, 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 network. Another challenge is the definition of an expressive and formal information model that enables the fine-grained description of virtual resources and facilitates information sharing between the various roles involved.

In this paper, we proposed a service oriented broker-based framework for resource description, publication, and discovery in virtual networking environments. The proposed framework relies on a novel service-oriented hierarchical business model as well as an expressive information model. The detailed architectural framework was presented, and its operation was illustrated using a REST-based virtualized content distribution scenario. Furthermore, a proof-of-concept prototype was implemented using a variety of technologies and tools, such as: Jersey, Grizzly Web server, JAXB, PostgreSQL, Vyatta virtual router, and the Xen Cloud Platform (XCP). A detailed performance analysis of the system was also presented.

Based on the conducted performance evaluation and comparative performance analysis, we can conclude that our proposed broker-based architecture yields acceptable performance (in terms of response time and network load) for the resources' publication, discovery, and negotiation operations, while incurring some significant delay for the virtual topology instantiation operation – a delay that is unavoidable due to the nature of the operation and is only incurred once, when a VNet is instantiated. When subjecting the system to variable loading conditions, we observed that the response time of the system shows a quadratic growth pattern for all three operations (publication, discovery, and negotiation), and a network load's logarithmic growth pattern for all operations, except the discovery/selection operation (in which a quadratic trend line was observed). As for the stress tests results, they demonstrated that the system's response time and network load increase in a quadratic fashion for the publication and the discovery/selection operations. We also observed that our resource brokerage system shows good scalability in terms of traffic handling, since it was tested for up to 15,000 requests (describing up to 120,000 resources) without failure. The deployment of the broker node in existing cloud environments would ensure even more scalability and resources' elasticity. Finally, when performing comparative performance testing, we found out that our broker-based architecture offers significant performance improvements in terms of response time (92.8% improvement) and incurred network load (77.3% improvement), when compared to a distributed brokerless architecture. Such performance improvement, combined with the reduced complexity of the resources' discovery operation enabled by the intermediary broker role, can contribute to an improved efficiency of the VN embedding process.

Abbreviations
VNs: Virtual networks; ISP: Internet service provider; SP: Service provider; VNP: Virtual network provider; NVE: Network virtualization environments; VNO: Virtual network operator; SLA: Service level agreement; MiCs: Micro clusters; MaC: Macro cluster; ADVNE: Aggregation-based discovery for virtual network environments; PDDR: Publication and dynamic discovery of resources; PIP: Physical infrastructure provider; VIP: Virtual infrastructure provider; SRR: Services and resources registry; RM: Resource manager; RAM: Resource allocation manager; VRDS: Virtual resource discovery and selection; RNE: Resource negotiation engine; RIC: Resource instantiation and configuration; SDT: Service deployment and testing; PMN: PIP management node; VMN: VIP management node; BN: Broker node.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
SR carried the design and implementation work related to the framework, and performed the performance evaluation of the solution. MEB is the lead of the research team, proposing the research topic and managing/ coordinating research activities. NK is an expert in virtualization and had contributions related to the information modeling and software architecture. RD and JP are experts in distributed systems and contributed with ideas related to the modeling and the performance evaluation. All authors read and approved the final manuscript.

Acknowledgements
This paper is an extended version of the article presented at IEEE WCNC 2013, under the title of "A Multi-Service Multi-role Integrated Information Model for Dynamic Resource Discovery in Virtual Networks".

Author details
[1]Lavasoft Software Canada Inc., 210-4700 rue de la Savane, Montréal, Quebec H4P 1T7, Canada. [2]College of Technological Innovation, Zayed University, Khalifa City B, P.O. Box 144534, Abu Dhabi, United Arab Emirates. [3]Department of Software and IT Engineering, University of Quebec, 1100 Notre-Dame West, Montréal, Quebec H3C 1K3, Canada. [4]Concordia Institute for Information Systems Engineering, Concordia University, 1515 St. Catherine W., Montréal, Quebec H4G 2W1, Canada. [5]Faculty of Engineering and Computer Science, Concordia University, 1515 St. Catherine W., Montréal, Quebec H4G 2W1, Canada.

Received: 8 September 2014 Accepted: 26 January 2015
Published online: 24 February 2015

References
1. Chowdhury NMMK, Boutaba R (2009) Network virtualization: state of the art and research challenges. IEEE Communications Magazine 47(7):20–26
2. Martin D, Vlker L, Zitterbarta M (2011) A flexible framework for future Internet design, assessment, and operation. Elsevier Computer Networks 55:910–918
3. Anderson T, Peterson L, Shenker S, Turner J (2005) Overcoming the Internet impasse through virtualization. IEEE Comput Mag 4(38):34–41
4. A. Bavier, N. feamster, M. Huang, L. Peterson, and J. Rexford, "VINI Veritas: Realistic and controlled network experimentation," In Proceedings of SIGCOMM'06, ACM Press, New York, USA, pp. 3–14
5. G. Schaffrah, 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 SIGCOMM 1st ACM workshop on Virtualized infrastructure systems and architectures (VISA 2009), ACM Press, New York, USA, pp. 63-72.
6. J. S. Turner, and D. E. Taylor, "Diversifying the Internet," In Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'05), IEEE Press, St. Louis, MO, USA, pp. 1-6.
7. Houidi I, Louati W, Zeghlache D, Baucke S (2009) "Virtual Resource Description and Clustering for Virtual Network Discovery". In: Proceedings of the IEEE International Conference on Communication., pp 1–6
8. Lv B, Wang Z, Huang T, Chen J, Liu Y (2010) "Virtual Resource Organization and Virtual Network Embedding Across Multiple Domains". In: Proceedings of International Conference on Multimedia Information Networking and Security., pp 725–728
9. Xu Y, Han Y, Niu W, Li Y, Lin T, Ci S (2012) A Reference Model for Virtual Resource Description and Discovery in Virtual Networks. In: Proceedings of ICCSA. Springer, Brazil, pp 297–310
10. H. Amarasinghe, A. Belbekkouche, and A. Karmouch, "Aggregation-based discovery for virtual network environments", In Proceedings of the IEEE International Conference on Communications (ICC 2012), IEEE Press, Ottawa, ON, pp. 1276–1280.
11. El Barachi M, Kara N, Dssouli R (2010) "Towards a Service-Oriented Network Virtualization Architecture,". In: Proceedings of the 3rd ITU-T Kaleidoscope Event 2010 (K-2010)., pp 1–7
12. El Barachi M, Rabah S, Kara N, Dssouli R, Paquet J (2013) "A Multi-Service Multi-Role Integrated Information Model for Dynamic Resource Discovery in Virtual Networks,". In: Proceedings of the IEEE Wireless Communications and Networking Conference 2013 (WCNC 2013)., pp 4777–4782
13. Feamster N, Gao L, Rexford J (2007) How to lease the internet in your spare time. SIGCOMM CCR 37(1):61–64
14. Niebert N, Khayat IE, Baucke S, Keller R, Rembarz R, Sachs J (2008) Network virtualization: a viable path towards the future Internet. Wireless Communication Journal 45(4):511–520
15. H. Medhioub, I. Houidi, W. Louati and D. Zeghlache, "Design, implementation and evaluation of virtual resource description and clustering framework", 25th IEEE International Conference on Advanced Information Networking and Applications (AINA 2011), IEEE Press, Biopolis, 83–89.
16. Houidi I, Louati W, Zeghlache D, Papadimitriou P, Mathy L (2010) Proceedings of the ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures., pp 41–48
17. J. Lakhani, P. Kumar, "Resource Selection Strategy Based on Propagation Delay in Cloud," In Proceedings of the International Conference on Communication Systems and Network Technologies (CSNT 2012), IEEE Press, Rajkot, pp. 11–13.
18. Yu M, Yi Y, Rexford J, Chiang M (2008) Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Comput Commun Rev 38(2):16–29
19. N. Malarvizhi and V. R. Uthariaraj, "A broker-based approach to resource discovery and selection in Grid environments," In Proceedings of the International Conference on Computer and Electrical Engineering (ICCEE 2008), IEEE Press, Phuket, pp. 322-326.
20. Kumar P, Pramanik PKD (2012) "Host selection methodology in cloud computing environment". International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 1(8):1–5
21. Zaheer, F.E.; Jin Xiao; Boutaba, R., "Multi-provider service negotiation and contracting in network virtualization," In Proceedings of the IEEE Network Operations and Management Symposium (NOMS 2010), IEEE Press, Osaka, pp.19-23.
22. Serhani M, Dssouli R, Hafid A, Sahraoui H (2005) A QoS broker based architecture for efficient web services selection. Proceedings of ICWS 2005:113–120
23. Richardson L, Ruby S (2007) "RESTful Web Services", O'Reilly & Associates, ISBN 10: 0-596-52926-0
24. "Jersey," [Online]. Available: http://jersey.java.net/. [Accessed January 2013].
25. "Project Grizzly," [Online]. Available: http://grizzly.java.net/. [Accessed January 2013]
26. "JAXB Project," [Online]. Available: https://jaxb.java.net/. [Accessed January 2013]
27. "eXist-db Project," [Online]. Available: http://exist-db.org/. [Accessed January 2013]
28. "PostgreSQL Global Development Group," [Online]. Available: http://www.postgresql.org/. [Accessed 15 February 2013]
29. "Xen Cloud Platform," Xen Project, [Online]. Available: http://www.xen.org/products/cloudxen.html. [Accessed 13 March 2013]
30. "Vyatta," Brocade, [Online]. Available: http://www.vyatta.com/. [Accessed January 2013]
31. "JMeter™," Apache Software Foundation, [Online]. Available: http://jmeter.apache.org/. [Accessed March 2013]