El Barachi et al. Journal of Internet Services and Applications (2015) 6:3
DOI 10.1186/s13174-015-0019-3

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

An open virtual multi-services networking architecture for the future internet

May El Barachi[1*], Nadjia Kara[2], Sleiman Rabah[3] and Mathieu Forgues[4]

Abstract

Network virtualization is considered as a promising way to overcome the limitations and fight the gradual ossification of the current Internet infrastructure. The network virtualization concept consists in the dynamic creation of several co-existing logical network instances (or virtual networks) over a shared physical network infrastructure. We have previously proposed a service-oriented hierarchical business model for virtual networking environments. This model promotes the idea of network as a service, by considering the functionalities offered by different types of network resources as services of different levels – services that can be dynamically discovered, used, and composed. In this paper, we propose an open, virtual, multi-services networking architecture enabling the realization of our business model. We also demonstrate the operation of our architecture using a virtualized QoS-enabled VoIP scenario. Moreover, virtual routing and control level performance was evaluated using proof-of-concept prototyping. Several important findings were made in the course of this work; one is that service-oriented concepts can be used to build open, flexible, and collaborative virtual networking environments. Another finding is that some of the existing open source virtual routing solutions such as Vyatta are only suitable for building small to medium size virtual networking infrastructures.

Keywords: Network virtualization; Future Internet; Service-oriented architecture; Virtual routing; Vyatta

1 Introduction

The concept of virtualization consists in the decoupling of physical resources from the service-level view, by adding an abstract layer (software), in between. The implementation of this concept gives the end-user the illusion of direct interaction with the physical resources, while allowing efficient utilization of resources/infrastructures and enhanced flexibility. Different forms of virtualization have been proposed, such as storage virtualization, server virtualization, application virtualization and more recently network virtualization. Storage virtualization refers to the separation of physical disk space from the logical assignment of that space, using various techniques (e.g. RAID and SAN). Server virtualization consists in the partitioning of the resources of a single physical machine into multiple execution environments (or virtual machines), each running its own operating system and server applications. Application virtualization refers to the isolation of a certain application from the operating system on which it runs, in order to achieve OS-independence and limit the effect of applications incompatibilities.

Network virtualization is an emerging concept that applies virtualization to entire networks. 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 [1]. Unlike Virtual Private Networks (VPNs) [2] that are limited to traffic isolation capabilities and do not allow customization nor administrative control, virtual networks (VNets) can potentially be built according to different design criteria and operated as service tailored networks.

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 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 [3]. By enabling a logical segmentation of the physical Internet infrastructure and the co-existence of heterogeneous virtual networking architectures on

* Correspondence: may.elbarachi@zu.ac.ae
[1]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

top of it, network virtualization is often seen as a cornerstone of the future Internet architecture [4].

Beyond the Internet's context, there are several important motivations behind the network virtualization concept. One of these motivations is the cost effective sharing of physical networking resources, by partitioning the resources of an existing infrastructure into slices and the allocation of these slices to different VNets (operated by different service providers). Another motivation is the potential for having customizable and service tailored networking solutions via the addition of new technologies or customized versions of existing technologies, in the virtualization layer.

Aiming to contribute to a future service-tailored Internet architecture, the main goal of this work is to propose and validate an open, service-oriented network virtualization platform for the future Internet. Our platform (dubbed the Open Virtual Playground) promotes the idea of "network as a service" by defining different levels of services to which networking resources are mapped, and which can be dynamically discovered, used, and composed. It relies on a novel service-oriented hierarchical business model [5,6] that introduces new business roles, and proposes the concept of vertical hierarchy between virtual network providers, as well as the concept of service building block and service reuse and composition.

The contributions of our work are of three folds: 1) It proposed a fine-grained, service oriented network virtualization architecture which encompasses the control functions needed for the instantiation, control, and management of virtual networks; 2) It details a concrete QoS-enabled VoIP scenario showcasing the architecture's operations, which include the REST-based interactions for the virtual networks' instantiation phase, and the SIP/COPS/MEGACO based interactions for the service invocation and usage phase; and 3) It discusses the implementation and performance evaluation of two prototypes related to the virtual data plane and the virtual control plane operation.

The rest of this paper is organized as follows: In section (The network virtualization concept: principles, goals, and motivating scenarios), we start with some background information about network virtualization and present two concrete use cases that could be enabled by this technology. In section (The open virtual playground architecture), we present an overview of our previously proposed business model, discuss the different components of our proposed Open Virtual Playground architecture, and illustrate its operation using a virtualized QoS-enabled VoIP scenario. This is followed by prototype-based performance evaluations, in section (Solution validation). We end the paper with a discussion of related work, before drawing our conclusions.

2 The network virtualization concept: principles, goals, and motivating scenarios

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. This topology is a subset of the underlying physical topology, in which 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 operated by a single entity, and virtual networks are logically isolated from each other. Several primitive forms of virtual networks have been proposed in the past, including virtual private networks (VPNs), overlay networks, and active/programmable networks. All these forms limit virtualization to a certain layer (e.g. application layer for overlay networks) and do not offer full administrative control and customization capabilities.

From an architectural perspective, network virtualization promotes several design goals [1], the most prominent ones being: the *coexistence* of multiple VNets (operated by different providers) within the same environment; *recursion and inheritance* between VNets allowing the nesting/creation of a VNet on top of another VNet (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 VNet; *manageability* allowing a provider to have full administrative control over a VNet; *isolation* between co-existing VNets to improve fault-tolerance, security and privacy; and *heterogeneity* of VNets as well as the physical infrastructures on which they rely.

There are several important motivations behind the network virtualization concept, including: *cost-effective sharing of physical networking resources*; *customizable networking solutions*; and the *convergence of existing network infrastructures*. Furthermore, several challenges must be addressed to enable the realization of this concept. Examples of these challenges include: the definition of *standard interfaces* between the different levels of the virtual networking hierarchy; the definition of *control functions related to the instantiation and configuration of virtual nodes/links*; ensuring *scalability* at the level of virtual nodes/links; enabling the *dynamic discovery of available physical/virtual resources*; the definition of *efficient global resource management strategies*; and the definition of suitable *business models and charging schemes* for virtual environments.

To illustrate the potential of the network virtualization concept, we now present two concrete use cases that could be enabled in virtual networking environments.

In order to define these use cases, we first analyzed the trends and proposed scenarios for future communication networks, such as the ones presented in the 4Ward project documents [7]. Afterwards, we focused on examining the limitations of the current Internet and on defining scenarios that would offer appealing services while addressing those limitations via virtualization.

Among the known areas of weakness of the current Internet architecture requiring innovation, we mention: stronger security; better mobility; more flexible routing; enhanced reliability; and better quality of service guarantees [8]. As for the categories of services studied, we chose the following categories: 1) VoIP as it represents one of the most important categories of applications offered in the current Internet; and 2) Context-aware smart applications (i.e. applications that are aware of the context surrounding them and capable of dynamically adapting to changing situations) as they are seen as one of the "killer" applications of the future.

2.1 VoIP service with two-dimensional QoS scheme

IP telephony is one of the important and popular application areas supported by the Internet. However, one of the main challenges faced by IP telephony is the inability to offer adequate quality of service guarantees to users, due to the inherent best effort nature of the current Internet architecture and the high cost of migration to new QoS technologies in a network as large as the Internet.

Our first use case illustrates how new QoS schemes could be introduced in existing networking infrastructures using the network virtualization concept. In this use case, a virtualized VoIP service with a two-dimensional QoS scheme is realized as follows: At the lowest level, we find the physical networking infrastructure (owned and managed by a Physical Infrastructure Provider), on top of which a first VNet (VN1) is instantiated and operated by a Virtual Infrastructure Provider. The latter deploys in VN1 a new QoS scheme enabling the differentiation between different classes of traffic (e.g. conversational, streaming, interactive, and background traffic classes), based on their resources requirements. Building on VN1 capabilities, a second VNet (VN2) is instantiated and deployed by a service provider offering sophisticated VoIP services and a session prioritization scheme enabling the distinction between sessions based on their level of importance from the user's perspective. This scheme, which constitutes a second QoS dimension, would enable the user to choose the appropriate class for each call (e.g. silver, gold, and platinum) based on its level of importance. We note that the value added in this scenario consists in the support of a two-dimensional QoS scheme and the support of sophisticated VoIP services. This value was added via the instantiated virtualization layers, without affecting the physical infrastructure, thus providing a smooth path for migration.

2.2 Context-aware value added services

Context-awareness is defined as the ability to use contextual (or situational) information to provide relevant information and/or services to the user [9]. Contextual information is usually collected using wireless sensor networks (WSNs).

Context-aware applications are an emerging category of intelligent applications that offer personalized services by adapting their behavior according to the users' needs and changing situation (e.g. personalized healthcare applications and smart shopping applications). These applications are seen as one of the "killer" applications of the future. The following scenario illustrates how such applications could be supported in a virtualized networking environment.

In this scenario, we find at the lowest level the PIPs, some of which are managing regular communication infrastructures, while others manage various types of WSNs used for the collection of different types of contextual information (e.g. spatial, physiological, and environmental data). On the top of these WSNs, we find VN1 that acts as a specialized network dedicated for the management of sensory information. Then, at the third level, we find a second VNet (VN2) leveraging the information management capabilities of VN1 as well as the communication capabilities of physical communication network to offer context-aware value added services to end users.

3 The open virtual playground architecture

In this section, we give an overview of our previously proposed network virtualization business model, and then present the Open Virtual Playground architecture.

3.1 Overview of proposed business model

Our previously proposed business model [6] is a service- oriented hierarchical model in which different levels of services (offered by various players) can be dynamically discovered, used, and composed. Figure 1 depicts our proposed model, in which four levels of services are defined, namely: 1) *Essential service*s constituting mandatory services needed for the basic operation of the network (i.e. routing/transport services); 2) *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); 3) *Service building blocks* acting as elementary services that can be used/combined to form more complex services (e.g. presence and call control); and 4) *End user services* constituting the value-added services offered to users.

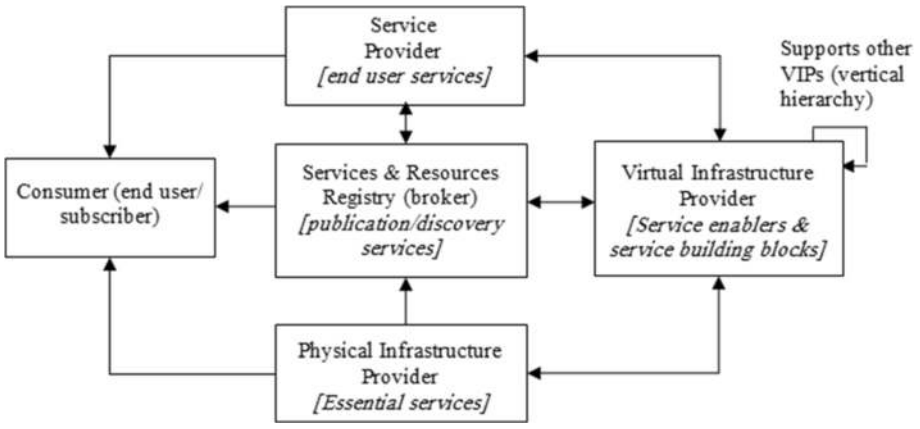

Figure 1 Proposed network virtualization business model.

Our business model defines 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 and aggregates virtual resources (offered by one or more PIPs), deploys any protocols/technologies in the instantiated VNet, 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; We envision three potential variations of the VIP role: a) A VIP that adds value in the virtualization layer by introducing a new technology or customizing existing protocols – the resulting VNet can be used by a SP to offer VAS running on it or resold to another VIP that leverages its capabilities to form another VNet on top of it (i.e. forming a vertical hierarchy); b) A VIP that uses virtualization to achieve interworking between heterogeneous physical infrastructures – the result being a unified network for others to use; and c) A VIP that implements more advanced services in the virtual layer to offer application building blocks that can be used by service providers to compose new value added services; 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 broker by providing information to find other parties and the services/resources they offer. This functional separation of roles enables the creation of an open and collaborative networking environment in which a rich set of resources and services are offered.

3.2 The proposed architecture

In this section, we present our proposed open virtual multi-services networking architecture, named the *Open Virtual Playground*. This architecture was designed to be *open* to different entities/roles/players and also open to change (i.e. dynamic and flexible), in addition to relying on *virtual*ization technology as a central concept in its operation, and offering a *playground* area for multi-players to interact, collaborate, and offer different services.

We start by presenting the overall architecture and its related functional entities and interfaces, and then present an illustrative session management scenario detailing its operation. The architecture was designed based on the defined business roles and the proposed functional split between them.

3.2.1 Functional entities and interfaces

As shown in Figure 2, the Open Virtual Playground architecture we are proposing is a layered architecture that introduces data and control planes at each of the three levels of the hierarchy. While the data plane provides essential data transportation functionality, the control plane encompasses all the control and management functions needed for the provisioning of different levels of services.

In order to realize the three main roles defined in our business model (i.e. PIP, VIP, and SP) and achieve the proposed functional split between them, three hierarchical levels are defined in our architecture, namely: 1) The physical network level (managed by the PIP); 2) the first virtual network level (managed by a VIP); and 3) the second virtual network level (managed by a SP). The consumer role accesses the services offered by the SP by interacting with the lowest level of the hierarchy (i.e. the physical network), while the SRR (i.e. the broker role) is

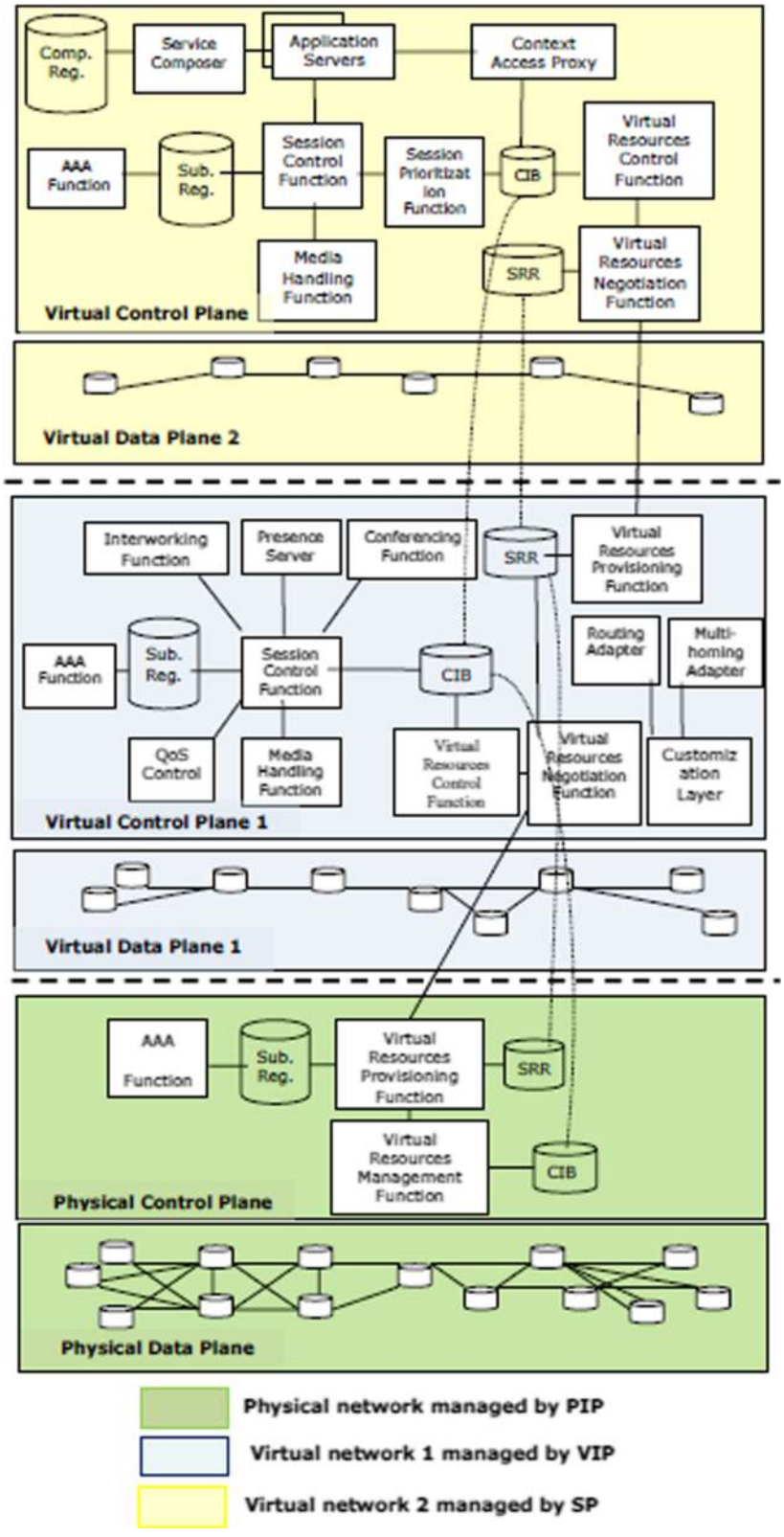

Figure 2 Proposed open virtual playground architecture.

a vertical role that interacts with all three levels of the hierarchy.

Similar to all business models, one entity could play several business roles at the same time. For instance a VIP could also play the role of a SP (i.e. offering both virtual topologies and value added services to end users). This situation would result in a two-tier architecture, encompassing two hierarchical levels (i.e. the physical network level and one virtual network level). While such model involves less interactions, it lacks the flexibility, openness, and role specialization achieved by the three-tier architectural model. In fact, maintaining the VIP as a separate entity offers the level of abstraction and separation between PIPs and SPs required to build virtual networks that are customized for particular service and user requirements. Furthermore, using a VIP as intermediary role simplifies the negotiation and virtual resource allocation process, in cases involving multiple SPs and multiple PIPs [10]. This process becomes much more complex when those SPs are directly interacting with multiple PIPs, in a fully distributed fashion (i.e. forming a full mesh topology). For the rest of this paper, we assume that each role is played by a separate entity, and focus on the three-tier hierarchical model which is described below.

At the *Physical Network Level*, we find the physical data plane containing regular and virtual routers connected to form the physical network infrastructure, as well as the physical control plane responsible for the following functions: Resource publication, resource negotiation, resource allocation and provisioning, and resource management. These functions are achieved using the following entities: The Services and resources registry (SRR) used for the publication/discovery of information about available resources; the Context information base (CIB) used for the management of contextual information related to the physical network (e.g. resources status and security level); the Subscription Registry containing all clients' subscription/authorization/authentication information; the AAA function using this information to authenticate, authorize, and charge VIPs for network resources utilization; the virtual resources provisioning function responsible for the negotiation of resources with VIPs, the allocation of virtual resources and the instantiation of virtual topologies; and the virtual resources management function responsible for the dynamic resource (re)allocation to VNets taking into consideration the resources status and the needs of VNets.

At the *First Virtual Network Level*, we find a virtual data plane encompassing a set of virtual nodes connected by virtual links (essentially a subset of the underlying physical topology), as well as virtual control plane 1. This latter encompasses the following functions: a set of service enablers and service building blocks, service publication, resource negotiation, resource discovery/selection, and service deployment/management. These functions are carried by the following entities: a SRR, a CIB, and a Sub. Reg. (playing similar roles to their peer entities in the lower layer, but in relation to VNet1 operation); a number of entities offering common support functions (e.g. session control, media handling, and interworking); a number of service building blocks (e.g. presence and conferencing); modules offering a customization of existing protocols (e.g. a content-based routing adapter and a multi-homing adapter); a virtual resources negotiation function used for the discovery and negotiation of resources (with PIP(s)) and the composition/instantiation of the VNet topology; a virtual resources control function responsible for the deployment of protocols and the operation of the VNet; and a virtual resources provisioning function used for the negotiation of virtual resources with other VIPs or SPs wishing to add another level in the hierarchy.

Similarly, the *Second Virtual Network Level* consists of a virtual data plane and a virtual control plane. The latter is responsible for the following: a set of end user services, end user service publication, resource negotiation, resource discovery/selection, service deployment/management, and service composition. We should mention that the SRR and the CIB repositories are distributed across all the levels of the hierarchy and a *cross-layered communication* between them enables the formation/maintenance of a global view of the physical/virtual networks contexts and available services/resources.

3.2.2 Illustrative session management scenario

To illustrate our architecture's operation, we now describe how the QoS-enabled VoIP scenario presented in section (VoIP service with two-dimensional QoS scheme) could be realized using it.

In this scenario, we find the following roles: a PIP managing the infrastructure offering communication capabilities; a VIP instantiating VN1 to offer QoS control, session control, and media handling as service enablers; and a SP instantiating VN2 to offer the VoIP value-added service, implementing a second QoS dimension (silver, gold, platinum), to consumers. Figure 3 illustrates the QoS-enabled VoIP service composition diagram, in terms of lower level sub-services, and how those sub-services are mapped onto physical networking resources.

As shown in Figures 3 and 4, the QoS-enabled VoIP scenario is divided into two phases: 1) the virtual networks' instantiation phase; and 2) the end-user service invocation and usage phase. Figure 4 depicts the virtual networks' instantiation phase that was realized using REST-based interactions between the different entities. REST is a network architectural style for distributed

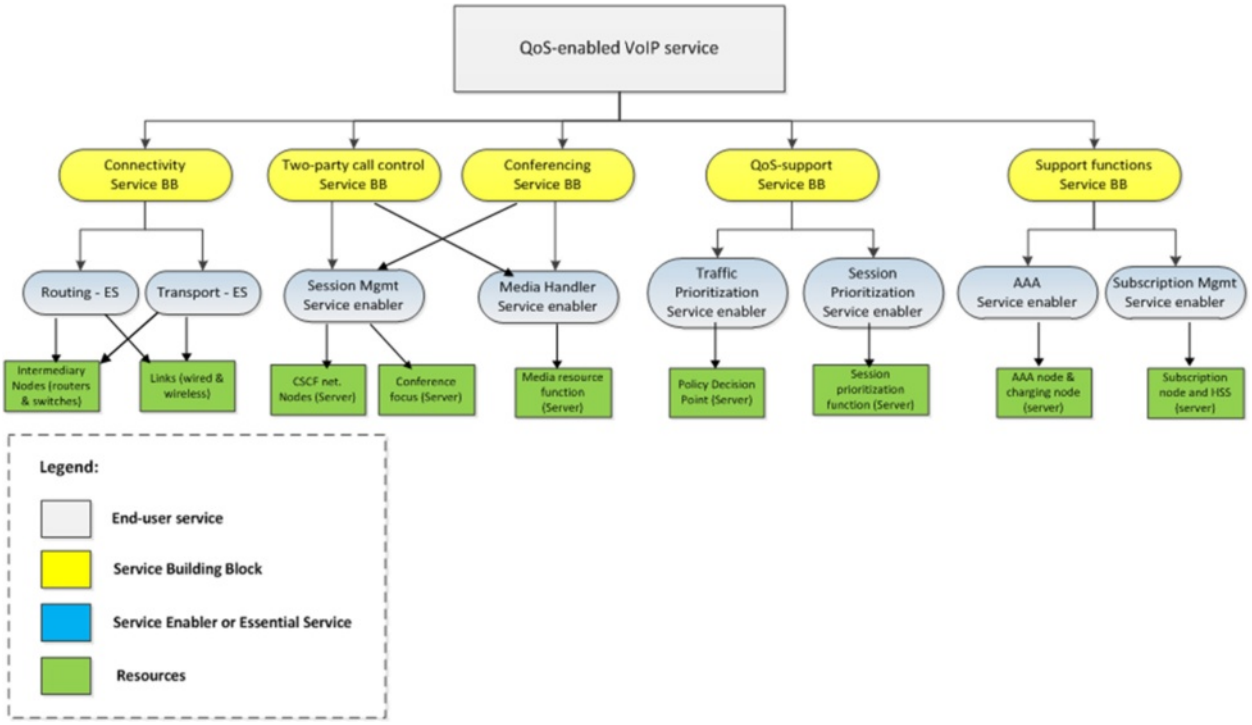

Figure 3 QoS-enabled VoIP service composition diagram.

hypermedia systems. There are several motivations behind our choice of REST-based interfaces, namely: its reliance on existing well known Web standards opens the door for various players to adopt such unified interfaces, for which the necessary infrastructure has already become pervasive; the fact that REST is simple, lightweight, and easy to develop; and the fact that it is resource-oriented and supports a wide range of resource description mechanisms.

Figure 5 illustrates the end-user service invocation and usage phase of the scenario. This phase was realized using three main protocols: SIP used for session control; COPS used for the exchange of policy-based resource allocation decisions; and MEGACO/H.248 used for the control of media handling nodes. We chose these protocols because of their extensibility and the fact that they provided the needed functionalities and are among the protocols supported in next generation networks.

As shown in Figure 4, the scenario starts when a PIP publishes (through its VRPF1) a description of the resources (step 1) it offers as well as their related constraints in a document that is used to populate the broker, using a POST request. In this request, the broker's resource creation service URI is specified. Once the resources' descriptions are created, a 200 OK message (step 2) is sent back to the PIP. In turn, the VIP (wishing to create VN1) sends a PIP discovery request (step 3) containing a document describing the resources to be leased, their desired availability, cost, and constraints. This request is sent using a GET message to the broker, which replies back (step 4) with a list of available providers that can satisfy the specified requirements. Upon receiving the PIPs list, the best PIP is selected by the VIP, using a selection/matching algorithm (step 5). Similarity-based matching algorithms such as the ones proposed in [11,12] can be used in this step. The VIP then sends a resource negotiation request (step 6), specifying the requested essential services and their constraints, to the selected PIP. After checking resources availability (step 7), the PIP replies with a resource negotiation response (step 8), specifying the offered resources and accepted constraints to the VIP, which concludes the negotiation process with a resource negotiation acknowledgement (step 9) confirming the negotiated resources and constraints. At this stage, the PIP carries a resource allocation and virtual topology instantiation process for VN1 (step 10), and sends an acknowledgement (step 11) of the topology instantiation to the VIP's VRNF, which is propagated to the VIP's VRCF (step 12). Afterwards, the VIP asks the PIP to deploy and test the specified service enablers (step 13), and gets a 200 OK message as reply (steps 15, 16). Once the service enablers are deployed and tested, the VIP's VRPF asks the broker to publish a description of the service enablers and their constraints (step 17), which in case of success results in a 200 OK message (step 18).

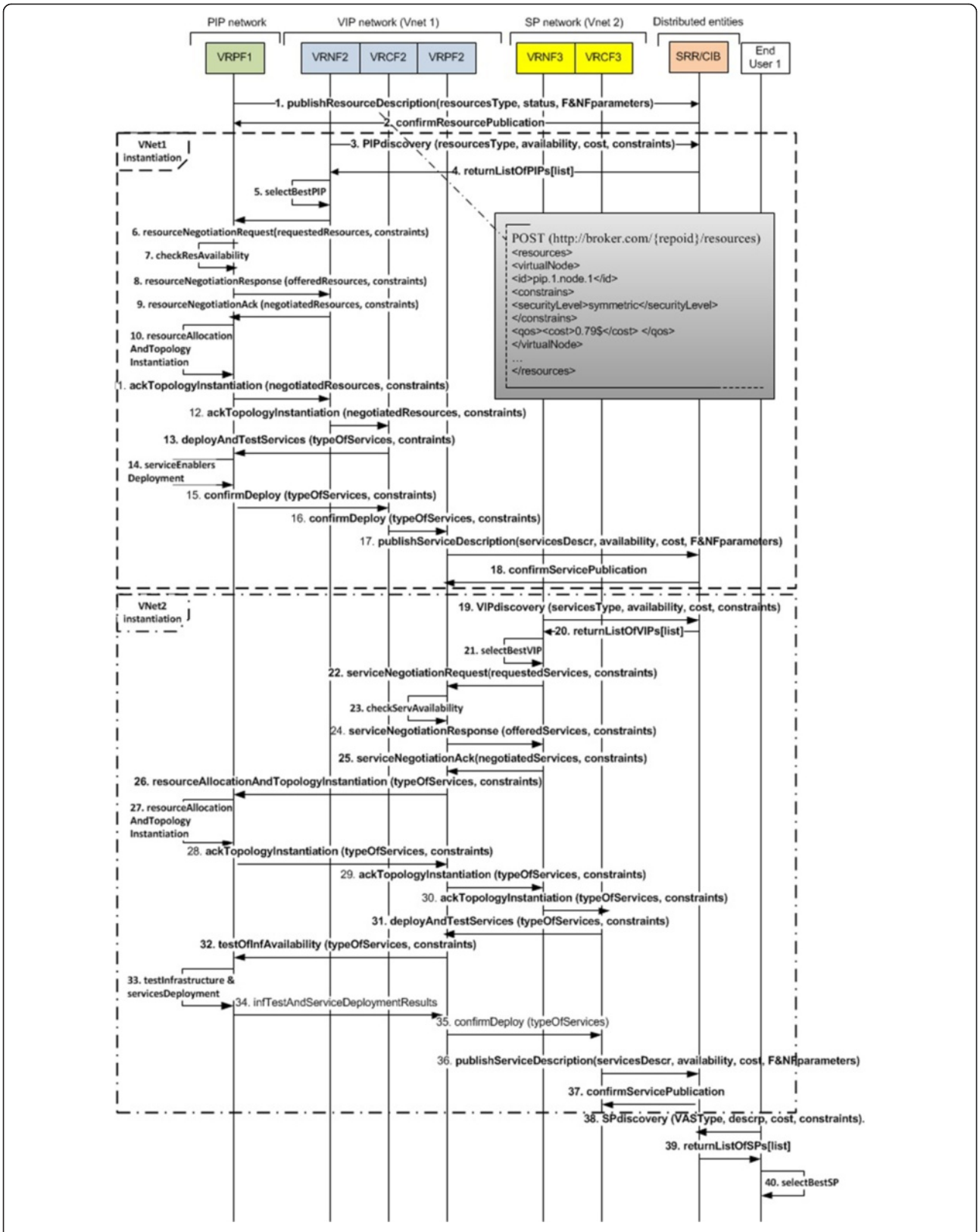

Figure 4 Illustrative session management scenario - Virtual Networks' instantiation phase.

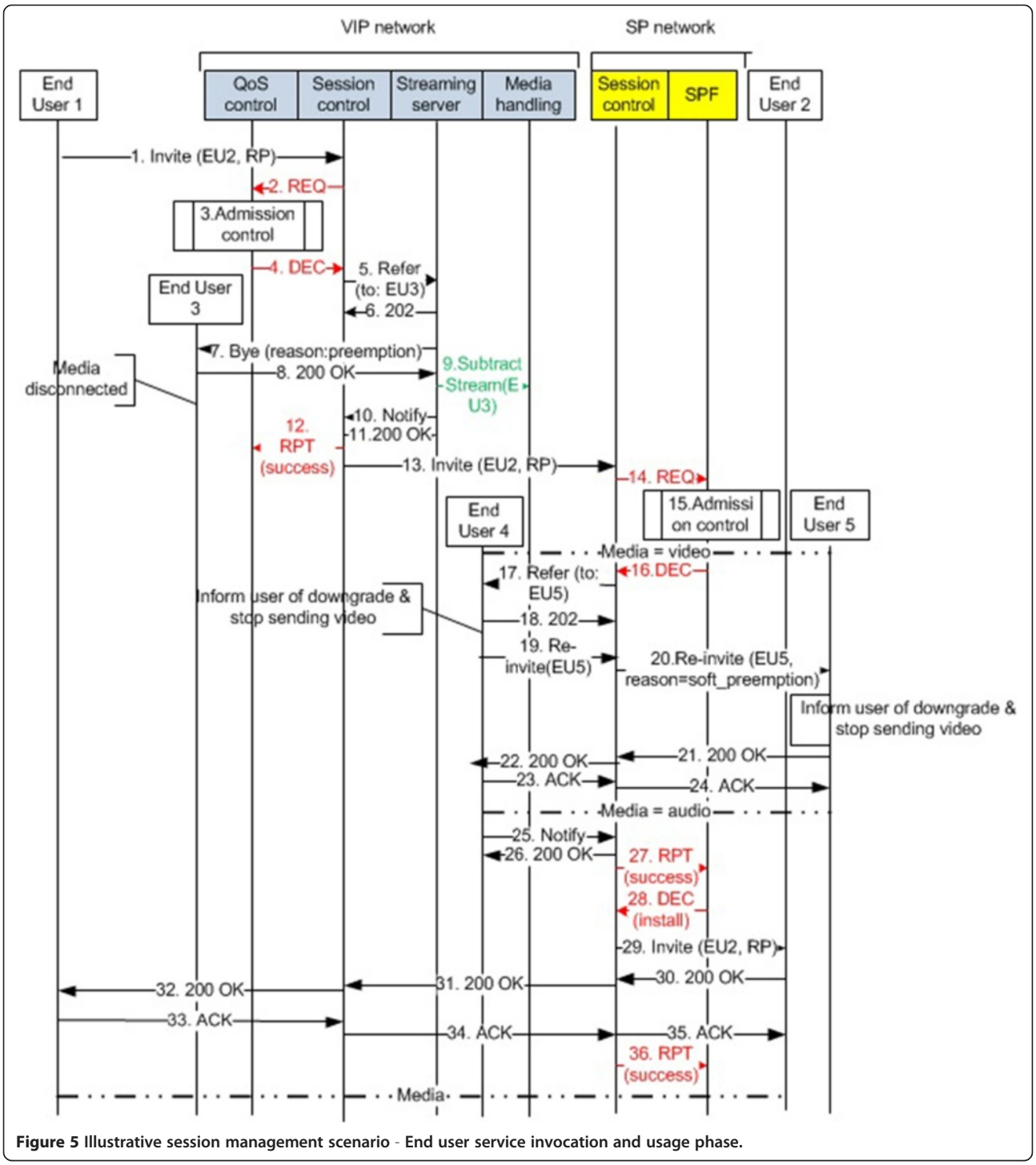

Figure 5 Illustrative session management scenario - End user service invocation and usage phase.

Meanwhile, a SP (wishing to create VN2) sends (using its VRNF) the broker a VIP discovery request (step 19) containing a document describing the service enablers to be used, their desired availability, cost, and constraints. The broker replies with a list of VIPs offering service enablers that comply with the request (step 20). Later, in step 21, the SP selects the best VIP to which he submits a service negotiation request (step 22). In steps 23 to 35, interactions related to service enablers' usage negotiation, VN2 topology instantiation, and the deployment of the QoS-enabled VoIP end user service offered by the SP are carried, similarly to the VIP::VN1 case. The main difference lays in the message parameters that refer to a different type of service in this case. When the

end user service is successfully deployed and tested, the SP sends its description to the broker (steps 36, 37). This description is then discovered (steps 38, 39) by the consumer that uses it to select the best SP (step 40).

Afterwards, the consumer (end-user 1) binds to the selected SP and invokes the QoS-enabled VoIP service, as depicted in Figure 5. In that case, end user 1 attempts to initiate a VoIP session of a certain category (e.g. platinum) with end user 2 by sending a SIP INVITE message, carrying a resource priority (RP) header (set according to the session category) to the session control function in the VIP network. In order to allocate resources to the call, the session control function sends the QoS control function (in the same network) a call admission request using a COPS REQ message (including the session information). After running a admission control algorithm (enabling the distinction between different classes of traffic), the QoS control function determines that an ongoing media streaming session must be terminated in order to free resources for the new VoIP session to be established. Therefore, the QoS control function sends a "trigger_termination" decision (using a COPS DEC message) to the session control function in relation to the (previously admitted) media streaming session between end user 3 and the streaming server. The session control function then sends a SIP REFER message instructing the media streaming server to terminate the session it has with end user 3. The server carries this instruction by sending a SIP BYE message, containing a reason header with the value "preemption", to end user 3, as well as a MEGACO subtractStream instruction to the media handling function. After the streaming session is terminated successfully, the streaming server sends a notification to the session control function (using a SIP NOTIFY message). This last returns a COPS RPT message indicating that it has enforced the QoS control function decision, then forwards the initial INVITE message to the session control function in the SP network.

In VNet2, similar interactions occur to enforce the second QoS dimension, in which differentiation between different classes of VoIP sessions (e.g. silver, gold, and platinum) is achieved. For instance, the session control function in the SP network interacts with the Session prioritization function (SPF) to admit the call in this VNet. In this case, a decision is made to downgrade an ongoing call (from video to audio) before admitting the new call. Therefore, a downgrade instruction is sent to end user 4, which carries it by sending a SIP re-INVITE to end user 5 (containing "audio" as new media type), thus renegotiating the session parameters. After the session is successfully downgraded, a COPS RPT (report state) message is returned to the SPF, which then authorizes the establishment of the VoIP session between end users 1 and 2.

It should be noted that the two proposed QoS schemes could potentially be supported using one virtual networking layer. In that case, both the QoS control function (needed for traffic prioritization) and the SPF (needed for session prioritization) must be implemented within this virtual network layer, which could add additional cost and complexity. On the other hand, implementing the two QoS schemes in two different virtual layers showcases the ability of a virtual network to build on the capability of another virtual network, in order to offer more advanced services to end users. This recursion and nesting capability could lead to a vertical hierarchy of virtual networks and advanced multi-tier architectures.

4 Solution validation

Two prototypes were built in order to validate our solution. The first prototype focused on the implementation and testing of the virtual data plane, while the second prototype focused on the realization and performance evaluation of the virtual control plane. For simplicity, we combined the roles of VIP and SP, and used a scenario with two layers: Layer 1 (consisting of the physical network level), and Layer 2 (consisting of the first virtual network level).

4.1 Virtual routing platform prototype

In order to validate and test the operation of the virtual data plane, we implemented a virtual routing platform using the open source Vyatta router. This router can be used to build physical or virtual routing platforms for small to large size enterprises. It provides almost all of the routing functionalities offered by others routing products (e.g., Cisco, Juniper, Extreme Networks). The main advantage of the Vyatta routing solution is its ability to be executed as a virtual machine using different virtualization technologies such as Xen and VMWare. Furthermore, it offers APIs that ease the integration of the virtual routing platform on a top of physical infrastructures. In this project, it has been used to build a virtual routing platform in order to validate the operation of the virtual data plane in our architecture, and evaluate its performance. The virtual routing platform built is illustrated in Figure 6.

As shown in the figure, the network was split in two separated areas. Area 1 included routers of cities A, B and C while area 2 included routers of cities D, E and F. Each router of a city was connected to its corresponding host. These areas were connected via routers G and H. Two hosts were connected to routers of cities G and H (e.g., Hosts G1 and G2 were connected to Node G). For cities A to F, we used two routers: master and backup. Four servers, each with 36 GB of disk space capacity were used to support this platform. Two HP servers were used to implement servers 1 and 2, each with

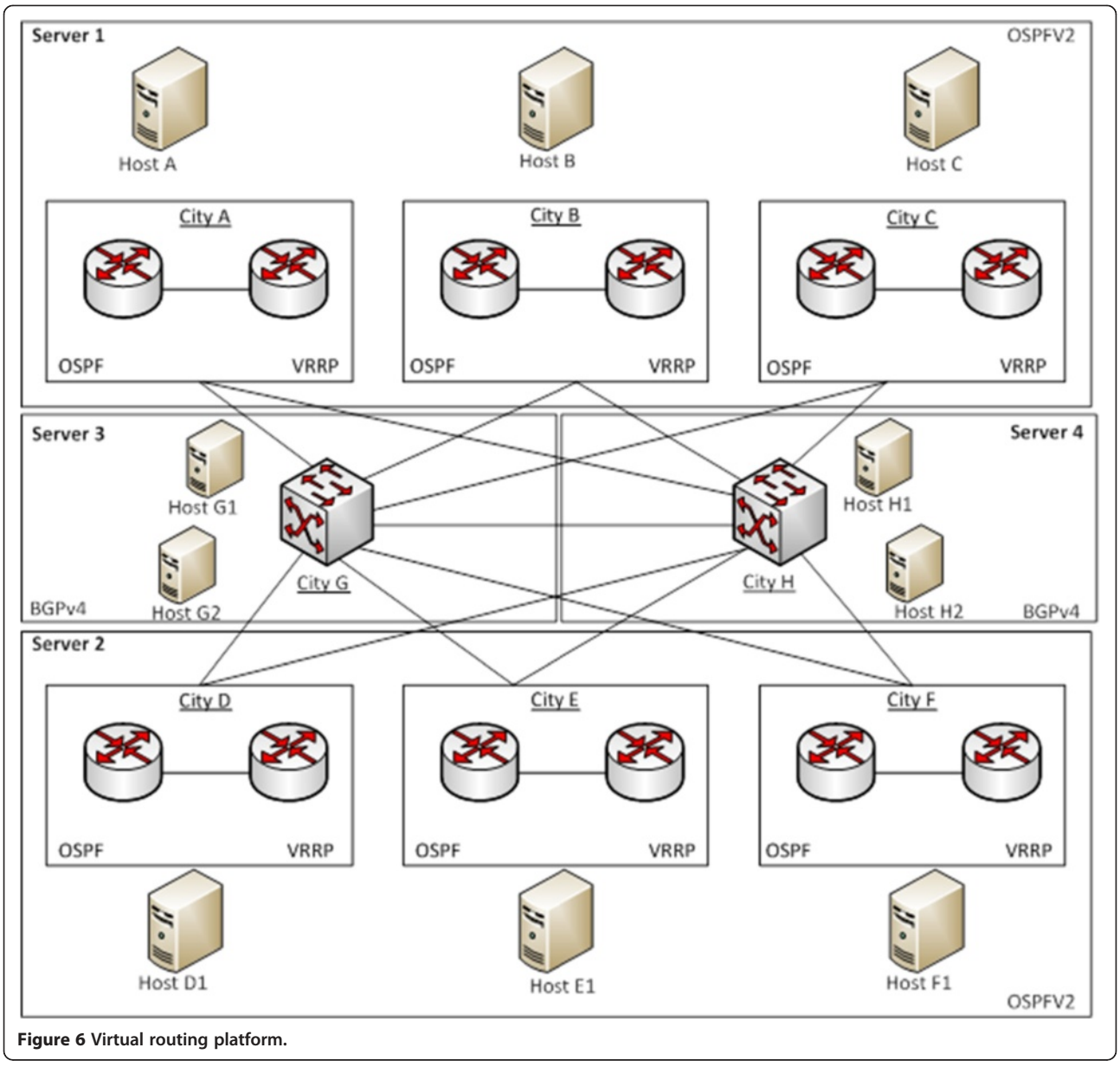

Figure 6 Virtual routing platform.

2x3.183GHz CPU and 3 Network interface Cards (NICs). Two Compaq servers hosted servers 3 and 4, each with 5 NICs. Server 3 had a 2x1.396 GHz CPU, while server 4 had a 1x1.396 GHz CPU. Each router was implemented using Vyatta version 6.3 as a virtual machine, with 1 CPU, 512 MB of memory, 4 GB of disk space and 3 NICs for cities A to F and 4 NICs for cities G and H. The main functionalities supported on each router were:1) OSPFv2 (servers 1 and 2) and BGPv4 (servers 3 and 4); 2) static routing, route redistribution between BGP and OSPF; 3) Static and dynamic (DHCP) IP address allocation; 4) Ethernet and VLAN (802.1Q) encapsulation; 5) Virtual Routing Redundancy Protocol (VRRP), CLI and SSHv2 administration and authentication; 6) Netflow Syslog and SNMPv2c diagnostics protocols. The hosts of cities A to H were implemented on virtual machines, each with 1 CPU, 256 MB of memory, 20GB disk space and 2 NICs. They were used to install a set of test tools to generate traffic across this network and to measure a set of performance metrics, namely: *JPerf* used to measure the data rate between two hosts using four TCP connections; *PingInfoView* used to measure delay in milliseconds taken by a router to answer a network request; *Ping* used to measure packet loss; *traceroute*/*pathping* used to determine the route taken by a packet from source to destination; *ping and timer* to determine the time taken for routing load balancing; and *Ostinato traffic generator* used to measure the load in packets/sec on processors, memories, disks of

the servers hosting the virtual routers. The technical specifications of the different platform components are summarized in Table 1 below.

Among the experiments conducted, we generated 4 TCP connections, each of 20 Mbps, using the JPerf tool, on each host. Each generated traffic was sent from one host to the other hosts in this network (e.g., from host A to host B, C, D, E, F, G and H). Then, using JPerf, we measured the data rate between the Hosts. Table 2 shows the data rate in Mbps between hosts A to H.

This performance shows that data rates in a virtual environment are comparable to those obtained in a traditional physical routing platform.

We also measured the delay taken by a router to answer to a network request and the packet loss. Therefore, we performed stress test using the traffic generator Ostinato which was installed on the hosts A to H as well as on hosts G1, G2, H1 and H2. Ostinato was configured to generate 10 data streams, each of 1000 packets/second, that are sent one after the other and looping back from the first stream. These streams were sent from each host to the other hosts (e.g., from host H to host A, B, C, D, E, F and G). The generated traffic yields to 6293 MHz, 6279 MHz, 2548 MHz and 1393 MHz of CPU usage as well as 2.02 GB, 2.04 GB, 1.29 GB and 951 MB of memory usage for server 1, 2, 3 and 4 respectively. This corresponds to almost 99%, 99%, 91% and 100% of total CPU capacity usage and almost 35%, 35%, 43% and 32% of total memory capacity usage for server 1 and 2, 3 and 4 respectively. Using the PingInfoView tool, we measured the delay and the packet loss. These performances are summarized in Table 3.

A significant packet loss and almost 200 ms of delay were noticed on Host H1 and H2. These behaviors were due to the fact that server 4 has a smaller CPU capacity than server 3 and the traffic loads that yielded to 100% of CPU capacity usage. Therefore, it is important to provide enough CPU capacity in the network environment under heavy traffic load.

Moreover, using PathPing, we analyzed the route taken by packets from source (e.g., Host B) to destination (e.g., host D). We noticed that packets took the appropriate routes. For instance, for PathPing from Host B to Host D, packets went through router G with a delay of almost 1 ms and no packet loss. The number of hops between routers was varying from 1 to 3. This allowed us to validate that the routing protocols (BGP and OSPF) were well configured. Moreover, using Ping and a timer, we measured the fault tolerance recovery time. For instance, we sent Ping message from Host D to router D. We turned off master router D and backup router D replaced it. We computed the difference between the timestamp of the last packet send by the master router and the timestamp of the first packet send by the backup router. We noticed that the recovery time was almost of 6 seconds and only one packet was lost.

As a conclusion, the BGP and OSPF routing protocols, the Ethernet and VLAN encapsulation as well as fault tolerance functionality are well supported by virtual routing platform using existing virtualization technologies. Moreover, the performance of a routing infrastructure built in a virtual environment using virtual machines is almost comparable to the routing performance in a physical network.

Table 2 Bandwidth measurements (Mbps)

	A	B	C	D	E	F	G	H
A	X	55.3	55.3	77	83.2	76.2	84.8	59.6
B	55.1	X	55.8	81	79.3	82.9	84.7	62
C	56.3	55.8	X	81.1	81.4	79.5	85.2	59.2
D	77.2	77.7	82.1	X	54.1	53.4	77.2	65.1
E	82.2	79.4	76.4	55.8	X	52.7	75.8	63.7
F	76.6	80.1	78.1	51.8	52.5	x	78	64.7
G	82.3	82.8	83.6	83.6	85.4	83.6	x	67
H	58.3	53.2	55.7	54.6	58.6	58.3	56.2	x

4.2 Virtual control plane prototype

As a second validation phase for our proposed solution, we focused on the implementation and testing of the control functions needed for the instantiation, control, and management of virtual networks. In this prototype, only a subset of the components proposed in section (The proposed architecture) was implemented. Furthermore, for

Table 1 Virtual routing platform specifications

Prototype component	Technical specification
Server 1	HP server with 2x3.183GHz CPU, 3 NICs, and 36 GB of disk space
Server 2	HP server with 2x3.183GHz CPU, 3 NICs, and 36 GB of disk space
Server 3	Compaq server with 2x1.396 GHz CPU, 5 NICs, and 36 GB of disk space
Server 4	Compaq server with 1x1.396 GHz CPU, 5 NICs, and 36 GB of disk space
Virtual Routers - for cities A to F	Vyatta version 6.3 as a virtual machine, with 1 CPU, 512 MB of memory, 4 GB of disk space and 3 NICs
Virtual Routers - for cities G and H	Vyatta version 6.3 as a virtual machine, with 1 CPU, 512 MB of memory, 4 GB of disk space and 4 NICs
Hosts of cities A to H	Running on virtual machines, each with 1 CPU, 256 MB of memory, 20GB disk space and 2 NICs

Table 3 Delay and packet loss measurements

Host A	Average ping time (ms)	Succeed count	Packet lost (%)
Host A	0	10	0
Router A	7	10	0
Router G	9	10	0
Router E	10	10	0
Router F	10	10	0
Host G1	10	10	0
Router D	11	10	0
Host G2	11	10	0
Router B	13	10	0
Host E	16	10	0
Host B	17	10	0
Host F	21	10	0
Router C	22	10	0
Host C	26	10	0
Host D	34	10	0
Router D	80	10	0
Router E	80	10	0
Router F	80	10	0
Router B	86	10	0
Router C	93	10	0
Host H1	190	4	60
Router H	217	10	0
Host H2	217	3	70

the brokerage node (the SRR/CIB), we opted for a centralized design in this first stage of implementation. Figure 7 depicts the software architecture of the implemented prototype and the technologies used.

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 encompasses a repository that contains resource related information and hosts the application logic realizing 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 providing 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). Reference [13] can be consulted for a detailed description of our proposed information model.

We selected Jersey [14], an open source JAX-RS (JSR 311) reference implementation, to implement the REST interfaces, and Grizzly web server [15] to deploy the web services. Moreover, we used JAXB 2 [16] for marshaling and un-marshaling of 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) and operating system type, virtualization environment in separate columns, whereas the XML document containing the resource description is stored as is in the same table. 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 discovery request contains two parts: the selection parameters (e.g. OS type, node type, virtualization environment); and the 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 discover 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.

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 as well as monitors allocated resources and updates their registered 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 the management and control of the substrate resources.

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 used the open source RDBMS PostgreSQL [17] 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

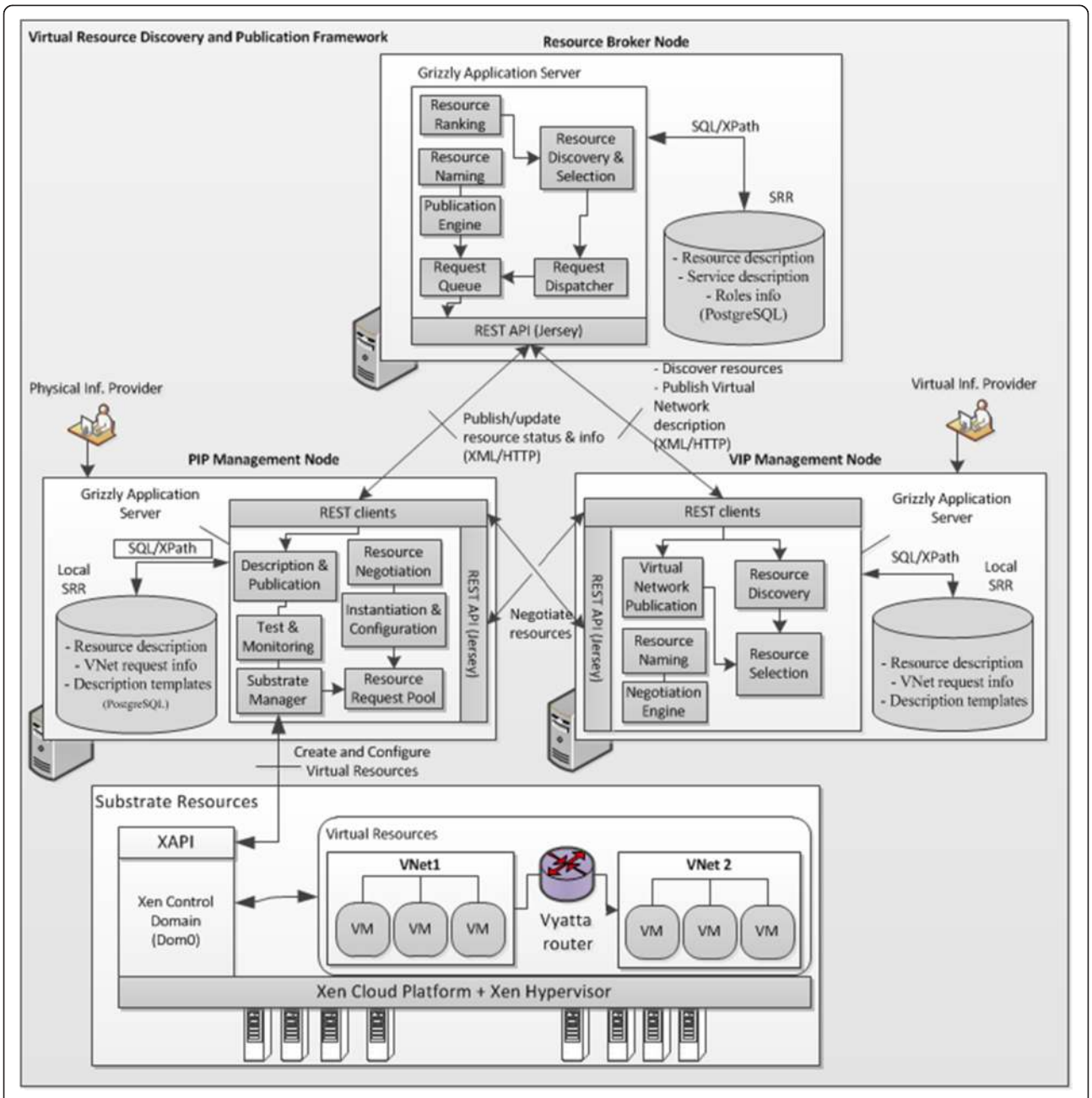

Figure 7 The virtual control plane prototype's software architecture.

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 documents as is 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 already the type of resource.

For the virtualization of substrate nodes, we used the Xen Cloud Platform (XCP) [18] that includes the Xen Hypervisor as well as Xen API (Xen Management API or XAPI). Based on Para-virtualization principles, Xen has proven 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 Xen Server's SDK that is provided by Citrix. The SM is responsible for automatically instantiating a virtual topology as described in the VNet 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), the modification of 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 [19] used in our first prototype to connect two or more virtual networks.

4.2.1 Prototype setup and test scenarios

As shown in Figure 8, 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, 512 MB of RAM, 20GB of disk space, and 5Mbps links. In this setup, we run two to four VMs on the same node.

Prior to run the experiments, we have generated a set 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 4. 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

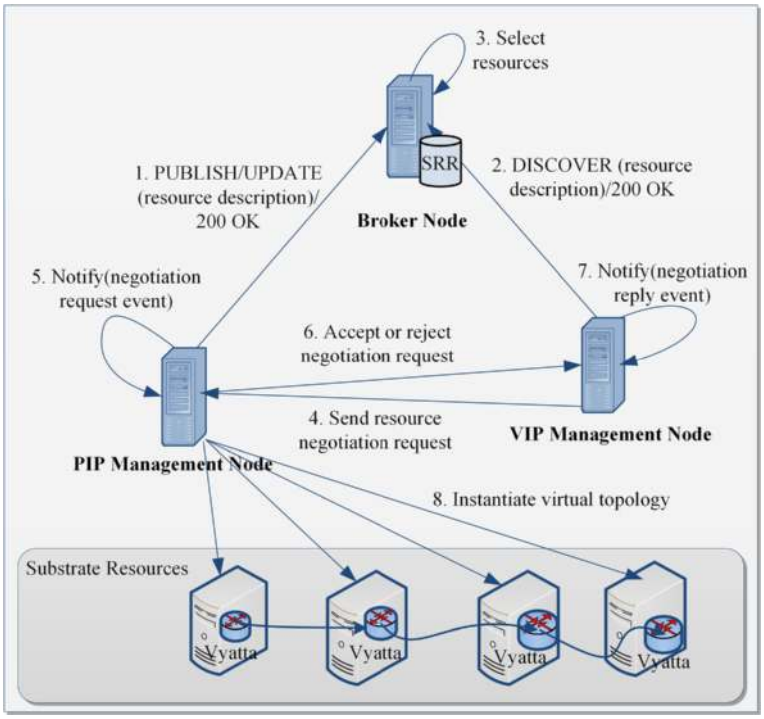

Figure 8 The virtual control plane prototype setup.

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 list of selected resources that best match the discovery request parameters, the VMN starts the resources' negotiation process by sending a negotiation request (containing a list of requested resources and related constraints) to the PMN. Upon receipt of the negotiation request by the PMN, a notification message is displayed on the PIP console. The negotiation process goes 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 9, 10, and 11 illustrate three of the screen shots of our prototype operation – namely the VIP resource discovery view, the PIP resource publication view, and the PIP virtual topology management view.

4.2.2 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 [20] 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 4 shows the evaluation results. Each result represents the mean value calculated over 20 trials.

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

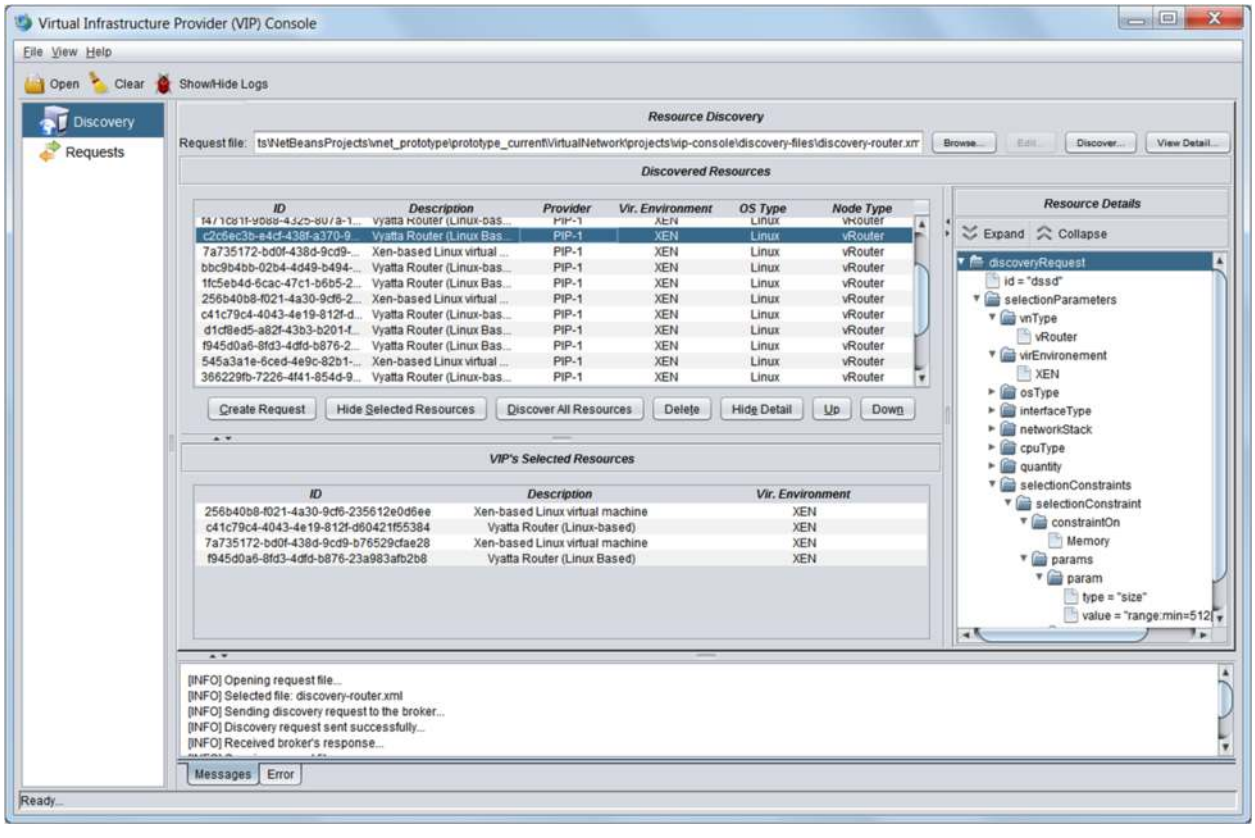

Figure 9 VIP resource discovery screenshot.

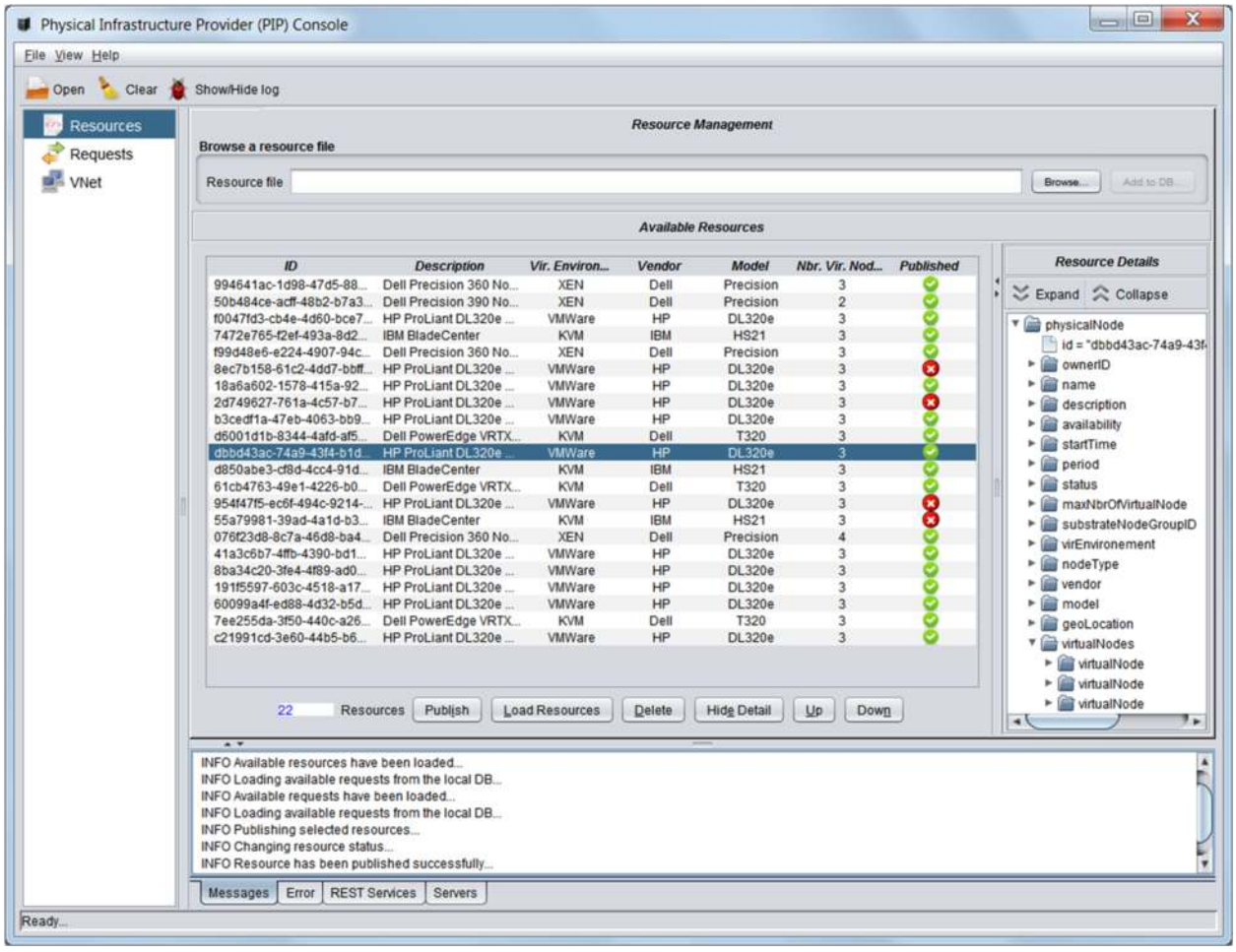

Figure 10 PIP resource publication screenshot.

repository. The results shown in the table are the average measurements over 20 trials. For each trial, we sent one resource publication request containing a document describing 2 virtual resources. On average, it took 245 ms to process this publication request, which generated 31.3 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 resources' database.

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 183 ms and 23.2 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 187 ms and 34.9 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

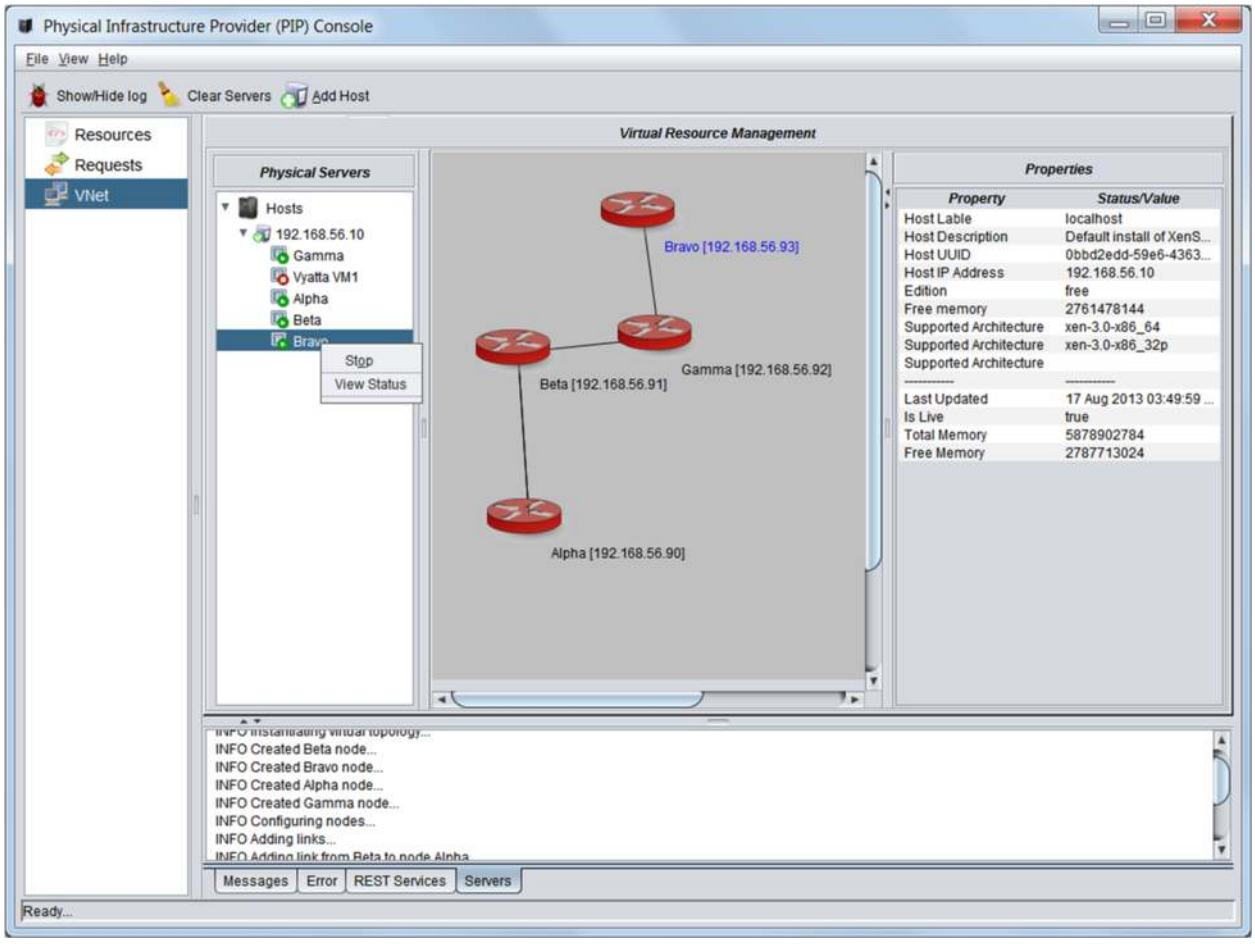

Figure 11 PIP virtual topology management screenshot.

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 11. On average, it takes one minute and 10 seconds to create and configure a Vyatta virtual machine, while it takes 5 minutes 55 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). 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. 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. It should be noted that the virtual topology instantiation time occurs during the first phase

Table 4 Network load and response time measurements

Operations	Interactions	Response time (ms)	Network load (KB)
Resource Publication [1 request/2 virtual resources published]	PMN – BN	245	31.3
Resource Discovery/selection [1 request/2 virtual resources discovered/50 resources processed during selection]	VMN – BN	183	23.2
Resource negotiation [1 request/2 virtual resources negotiated]	VMN – PMN	187	34.9
Virtual Topology instantiation [4 virtual routers, 3 virtual links]	PMN – substrate nodes	355000	142.2

(i.e. the virtual networks instantiation phase shown in Figure 4) and does not affect the performance of real-time end user services (e.g. VoIP) during their usage and operation (which constitutes the second phase shown in Figure 5). In fact, virtual networks are typically instantiated, the end user services are deployed on them at a later stage, when needed.

4.2.3 Scalability 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 12 and 13 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 an exponential fashion, while the network load increased in a linear fashion. This exponential increase in response time is due to several factors such as: Database overhead caused by reading/writing records; description documents marshaling and un-marshaling; and HTTP requests' processing overhead.

As for the discovery operation, the broker was successfully tested for up to 12,000 discovered resources, and the response time and network load both showed exponential growth patterns with respect to the number of discovered resources. 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.

5 Related work

Several management and control architectures have been proposed for virtualized networks. However, to the best of our knowledge, these proposals have not yet fully investigated a service-oriented hierarchical architecture, in which different types of functionalities that could be offered by a network resource (e.g., low level routing/transport functionalities, high level application logics) can be dynamically discovered, used and composed.

In [21], authors propose a layered network architecture based on four planes: data, management, control, and knowledge. These planes are implemented using different isolated virtual networks sharing the same physical infrastructure. This architecture enables the reuse of the current Internet's data plane and the deployment of concurrent next generation Internet that provides new

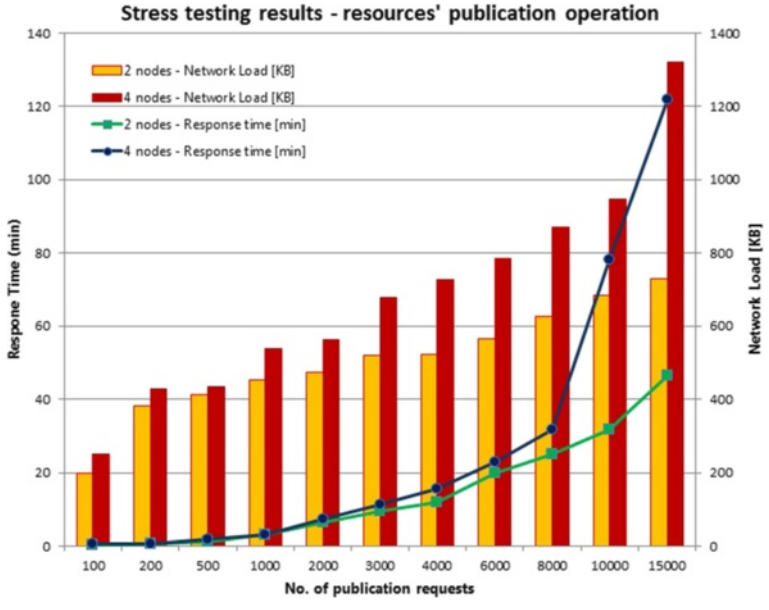

Figure 12 Stress testing results for resources' publication operation.

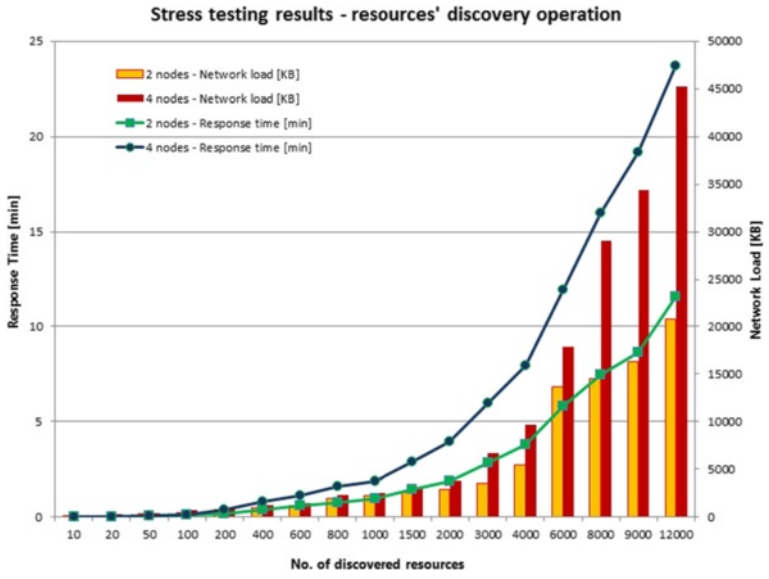

Figure 13 Stress testing results for resources' negotiation operation.

network functionalities. However, network resource discovery, usage and composition are not considered and the network partitioning proposed in [21] is far from what could be done by the architecture proposed in this paper.

A few architectures have been proposed for virtualized networks that support resources publication, discovery and access. One example is the service-oriented framework for composing network and cloud services [22]. That proposed framework is based on SOA architecture and virtualization in cloud computing [23] as well as the network virtualization business model described in [1]. The key drawback of this proposal is that a very coarse-grained service-oriented architecture is proposed with no detail on the service publication, composition, and discovery.

The Service-oriented MGON (MultiGranular Optical Network) [24] is fine-grained service-oriented network architecture that has been proposed for cloud environment. This solution provides a mapping strategy of application requests expressed in terms of QoS parameters (e.g., delay, bandwidth) to technology-specific attributes (e.g., burst size, wavelength, path) in order to offer different levels of service granularity to clouds. However, it proposes resource discovery strategy based on proprietary control and management interface with new packet format specifically tied to optical networks. Moreover, unlike the network virtualization environment where multiple heterogeneous networks can be shared by different service providers, MGON relies on a flat virtualized network architecture with a traditional ISP business model.

A hierarchical architecture for virtualized network has been proposed in [25]. It targets the virtual resource management issue and proposes a framework to discover, select and use network resources in order to build virtual networks. However, no detailed information is given on how resources are discovered and selected.

In reference [26], the authors discuss how SOA (Service Oriented Architecture) can be leveraged to realize the concept of Network as a Service (NaaS), thus enabling the convergence of virtual networking and cloud computing. Different technologies related to the realization of the NaaS concept are discussed in the paper, including network service description, discovery, and composition solutions. None of these solutions present an end-to-end architecture for network virtualization.

6 Conclusions and lessons learned

Network virtualization is an emerging concept that enables the dynamic creation of virtual networks over a shared physical network infrastructure. There are several motivations behind this concept, including increased flexibility, diversity, and manageability in networking environments. We have previously proposed a service-oriented hierarchical business model for virtual networking environments. This model aims at creating a dynamic and collaborative environment, in which a large pool of virtualized networking resources, which are seen as services (of different levels), can be dynamically discovered, used, and composed. In this paper, we proposed an open virtual multi-services networking architecture enabling the realization of our business model. Furthermore, we illustrated our architecture's operation using a virtualized QoS-enabled VoIP scenario. This paper also presented performance analysis of a virtual routing platform that was implemented using open source Vyatta router.

Furthermore, a proof-of-concept prototype of the virtual control plane 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). Basic performance measurements and stress testing results of the main operations were also collected and analyzed.

During the course of this project, we learned several important lessons. The first is that service-oriented concepts can be very useful for building open, flexible, and collaborative virtual networking environments, in which various network functionalities are considered as services that can be reused and composed, and roles are considered to be distributed and loosely coupled entities interacting via programmable interfaces. Another lesson we learned is that the Vyatta routing solution is suitable for small-size to mid-size network infrastructures with performance behaviors almost comparable to those of a physical network. A third lesson learned is that Vyatta routing solution is not adapted to large-size networks (e.g., backbone network). Indeed, we noticed that due to overwhelming data traffic sent through the platform, the Vyatta processor load increases very fast and therefore the processor of the hypervisor (in our case VMWare) quickly becomes saturated. Moreover, it may be difficult to run a virtual machine that provides a network interface with high data rates to ensure the transfer of higher traffic load. However, servers with greater capacities than those chosen in this paper should be considered to identify what would be the maximum traffic load supported by such platform.

Abbreviations

VPNs: Virtual Private Networks; VNets: Virtual Networks; WSNs: Wireless Sensor Networks; PIP: Physical Infrastructure Provider; VIP: Virtual Infrastructure Provider; SRR: Services and Resources Registry; CIB: Context Information Base; PMN: PIP Management Node; VMN: VIP Management Node; BN: Broker Node; RDS: Resource Discovery and Selection Engine; XCP: Xen Cloud Platform; SM: Substrate Manager; MGON: MultiGranular Optical Network; SOA: Service Oriented Architecture; NaaS: Network as a Service.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

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 business modeling and the architecture. SR carried the design and implementation work related to the virtual control plane prototype, and performed the related performance evaluation. MF Worked on the virtual routing platform prototype implementation and testing. All authors read and approved the final manuscript.

Acknowledgement

This paper is an extended version of the article presented at IEEE CCNC 2012, under the title of "Open Virtual Playground: Initial Architecture and Results".

Author details

[1]College of Technological Innovation, Zayed University, Khalifa City B, P.O. Box 144534, Abu Dhabi, United Arab Emirates. [2]Department of Software and IT Engineering, University of Quebec, 1100 Notre-Dame, West, Montréal, Quebec, H3C 1 K3, Canada. [3]Faculty of Engineering and Computer Science, Concordia University, 1515 St. Catherine W, Montreal, Quebec, H4G 2 W1, Canada. [4]Ericsson Canada, 8400 Blvd, Décarie Montréal, Québec, H4P 2 N2, Canada.

Received: 11 September 2014 Accepted: 22 December 2014
Published online: 28 February 2015

References

1. Chowdhury N, Boutaba R (2009) Network virtualization: state of the art and research challenges. IEEE Commun Mag 47(7):20–26
2. Rosen E and Rekhter Y (2006) "BGP/MPLS IP Virtual Private Networks (VPNs)," RFC 4364, Internet Engineering Task Force
3. Turner JS and Taylor "Diversifying the Internet". In: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'05), IEEE Press, St. Louis, MO, USA, pp. 1-6.
4. Anderson T, Perterson L, Shenker S, Turner J (2004) Overcoming the internet impasse through virtualization. In Proceedings of ACM HOTNETS, San Diego, CA, USA
5. El Barachi M, Kara N and 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.
6. El Barachi M, Kara N and Dssouli R (2012) "Open Virtual Playground: Initial Architecture and Results". In: Proceedings of the 9th IEEE Consumer Communications and Networking Conference 2012 (CCNC 2012), pp 576–581.
7. "4WARD – Architecture and Design for the Future Internet: Project-wide Evaluation of Business Use Cases". Public deliverable No. FP7-ICT-2007-1-216041-4WARD/D-1.2, European Union's 7th Framework, Sweden, December 2009.
8. Feldmann A (2007) Internet clean-slate design: What and why?". SIGCOMM CCR 37(3):59–64
9. Abowd G, Dey A, Brown P, Davies N, Smith M and Steggles P "Towards a Better Understanding of Context and Context-Awareness". In: Hans-Werner Gellersen (Ed) Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC '99), Springer-Verlag, London, UK, UK, pp 304-307
10. Hasam M, Amarasinghe H and Karmouch A "Network Virtualization: Dealing with multiple infrastructure providers". In: Proceedings of the IEEE International Conference on Communications (ICC 2012), IEEE Press, Ottawa, ON, Canada, pp 5890–5895
11. Medhioub H, Houidi I, Louati W and Zeghlache D (2011) "Design, implementation and evaluation of virtual resource description and clustering framework". 25th IEEE International Conference on Advanced Information Networking and Applications 2011, IEEE Press, Biopolis, pp 83–89.
12. Houidi I, Louati W, Zeghlache D, Papadimitriou P and Mathy L (2010) "Adaptive virtual network provisioning". In: Proceedings of the ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures, ACM Press, New York, NY, USA, pp 41-48.
13. El Barachi M, Rabah S, Kara N, Dssouli R and 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), IEEE Press, Shanghai, China, pp 4777–4782.
14. "Jersey," [Online]. Available: http://jersey.java.net/.
15. "Project Grizzly" [Online]. Available: http://grizzly.java.net/.
16. "JAXB Project" [Online]. Available: https://jaxb.java.net/.
17. "PostgreSQL Global Development Group" [Online]. Available: http://www.postgresql.org/.
18. "Xen Cloud Platform" Xen Porject, [Online]. Available: http://www.xen.org/products/cloudxen.html.
19. "Vyatta," Brocade, [Online]. Available: http://www.vyatta.com/.
20. "JMeter™" Apache Software Foundation, [Online]. Available: http://jmeter.apache.org/.
21. Jin D, Li Y, Zhou Y, Su L and Zeng L (2009) "A virtualized-based network architecture for next generation internet". In: Proceedings of the Third International Conference on Anti-counterfeiting, Security, and Identification in Communication, IEEE Press, Hong Kong, pp 58–62.
22. Qiang D (2011) "Modeling and performance analysis on network virtualization for composite network-cloud service provisioning". In: Proceedings of the 11th IEEE World Congress on Services, pp 548–555.

23. Zhang L-J and Zhou Q (2009) "CCOA: Cloud Computing Open Architecture", In: the Proceedings of the 1st Symposium on Network System Design and Implementation (NSDI '09), IEEE Press, Los Angeles, CA, USA, pp 607-616.
24. Zervas G, Martini V, Qin Y, Escalona E, Nejabati R, Simeonidou D, Baroncelli F, Martini B, Torkmen K, Castoldi P (2010) Service-oriented multigranular optical network architecture for clouds. IEEE/OSA J Opt Commun Netw 2 (10):883–891
25. Lv B, Wang Z, Huang T, Chen J and Liu Y "Hierarchical Virtual Resource Management Architecture for Network Virtualization". In: Proceedings of the 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM 2010), IEEE Press, Chengdu, pp 1–4
26. Duan Q, Yan Y, Vasilakos AV (2012) A survey on service-oriented network virtualization toward convergence of networking and cloud computing. IEEE Trans Netw Serv Manag 9(4):373–392