INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2010; **20**: 129–148
Published online 15 January 2010 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/nem.732

Specifying and Provisioning Virtual Infrastructures with HIPerNET

Fabienne Anhalt,*† Guilherme Koslovski, Pascale Vicat-Blanc Primet

{fabienne.anhalt,guilherme.koslovski,pascale.primet}@ens-lyon.fr
INRIA, LIP, ENS Lyon

SUMMARY

With the expansion and convergence of communication and computing, dynamic provisioning of customized networking and processing infrastructures, as well as resource virtualization, are appealing concepts and technologies. Therefore, new models and tools are needed to allow users to create, trust and enjoy such on-demand virtual infrastructures within a wide area context. This paper presents the HIPerNET framework we are designing and developing for creating, managing and controlling virtual infrastructures in the context of high-speed Internet. The key idea of this proposal is the combination of network- and system-virtualization associated with controlled resource reservation to provide fully isolated environments. HIPerNET's motivations and design principles are presented. Then we examine specifically how this framework handles the virtual infrastructures, called Virtual Private eXecution Infrastructures (VPXI). To help specifying customized isolated infrastructures, HIPerNET relies on VXDL, a language for VPXI description and modeling which considers end-host resource as well as the virtual network topology interconnecting them, including virtual routers. After the specification, allocation and scheduling phases, HIPerNET helps in provisioning, deploying and configuring virtual private execution infrastructures. This means, it triggers the dynamic configuration of all the equipments involved. In this paper we concentrate on network configuration, particularly to achieve network performance isolation. We also study and evaluate mechanisms to implement and configure virtual-link control. Experimental results obtained within the Grid'5000 testbed are presented and analyzed. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: IaaS, network virtualization, resource virtualization, VXDL, VPXI, provisioning, virtual infrastructure, isolation

1. INTRODUCTION

The expansion and convergence of computing and communication paint a new vision of the Internet: it is not only a black box providing pipes between edge machines, but becomes a world-wide reservoir increasingly embedding computational and storage resources to meet the requirement of emerging applications[1]. Consequently, the promising vision of grid computing—to bring together geographically distributed resources to build very large-scale computing environments for data- or computing-intensive applications—along with the service wave led naturally to the 'Infrastructure as a Service' paradigm[2].

High-end applications present new challenges—e.g., the amount of data to process, the distribution of the data (wide-spreading of data sources over the territory), the heterogeneity of the data (lack of well established standards), confidentiality—which results in high communication, storage, and computation requirements. Intensive data-processing applications require an access to infrastructures with high-performance data-movement facilities coordinated with computational resources. However, current

*Correspondence to: Fabienne Anhalt, INRIA, LIP—ENS Lyon, Lyon 69007, France
†E-mail: fabienne.anhalt@ens-lyon.fr

Received 20 February 2009
Revised 27 February 2009
Accepted 15 September 2009

grid or cloud computing solutions[3,4] built over traditional TCP/IP technology do not provide the performance isolation and predictability required by these users. Other applications need interconnections of large-scale instruments with high-performance computing platforms. The proposed best effort approaches often result in user dissatisfaction and/or resource under-utilization. Moreover, system and middleware heterogeneity and complexity posed barriers to application portability. We expect future Internet applications will be more and more demanding in terms of security, performance and quality of service. New technologies such as machine, links or router virtualization are now envisionned as solutions to these issues. Future applications may highly benefit from these innovations.

In this paper, we argue that the extension of the Infrastructure as a Service paradigm to network resources is promising and powerful. We show how it can be implemented through the Virtual Private eXecution Infrastructure (VPXI) concept. Seen as a fundamental building block, VPXIs could mark a step forward in transforming the Internet into a huge, shared computing and communication facility. Any entity (individual user, community, government, corporate customer, high-level service provider) would dynamically rent the IT resources combined with the networking resources it needs to solve its specific problem. Based on this vision, this paper presents HIPerNET, a framework to create, manage and control confined Virtual Private eXecution Infrastructures (VPXI) in a large-scale distributed environment.

In Section 2 we present the Virtual Private eXecution Infrastructure concept and the HIPerNET framework, highlighting the originalities of its design. Section 3 discusses the specification and allocation process of VPXIs based on the VXDL, a language for virtual-infrastructure specification and modeling which considers end-host resources as well as virtual network topology, including virtual routers and timeline. In Section 4 we highlight the configuration step. The configuration of a VPXI is based on the instantiation of virtual entities as well as the configuration of various embedded mechanisms. We focus here on the control of network performance and on the isolation aspect. In Section 5 we give experimental results obtained on the Grid'5000 testbed. Related works are in Section 6 while conclusion and perspectives are developed in Section 7.

2. HIPERNET CONCEPTS

2.1 Combining OS and Network Virtualization

The virtualization concept[5,6,7] enables an efficient separation between services or applications and physical resources. The virtual-machines paradigm is becoming a key feature of servers, distributed systems and grids as it provides a powerful abstraction. It has the potential to simplify resource management and to offer a great flexibility in resource usage. Each Virtual Machine (VM) a) provides a confined environment where non-trusted applications can be run, b) allows establishing limits in hardware-resource access and usage, through isolation techniques, c) allows adapting the runtime environment to the application instead of porting the application to the runtime environment (this enhances application portability), d) allows using dedicated or optimized OS mechanisms (scheduler, virtual memory management, network protocol) for each application, e) enables applications and processes running within a VM to be managed as a whole.

A virtual private network is classically provisioned over a public network infrastructure to provide dedicated connectivity to a closed group of users. Resource-guaranteed VPNs[8] can be obtained from a carrier but are generally static and require complex service level agreements (SLAs) and overheads. Tunneled VPNs [9] such as those in use over the Internet are more lightweight but offer low performance and no assured quality of service. The functionality required for the automating dynamic provisioning of lightpath or virtual private network capacities is emerging[10].

We propose here to combine and apply the concepts to both the IT resources and the network to enable the creation of multiple, isolated and protected virtual aggregates on the same set of physical resources by sharing them in time and space. These aggregations of virtual resources organized with virtual interconnections are called Virtual Private eXecution Infrastructures (VPXI). Moreovoer, virtualizing routers

DOI: 10.1002/nem

and switching equipments opens an interesting door for the customization of packet routing, packet scheduling and traffic engineering for each virtual interconnection network. Router-function customization as well as data-plane virtualization offers a high flexibility for each infrastructure. Therefore, conceptually, the virtual infrastructures are logically isolated by virtualization and enable the deployment of independant and customized bandwidth provisioning, channel encryption, addressing strategies. The isolation could also provide a high security level for each infrastructure. However, obtaining full performance and security isolation remains a key technical issue and most of the proposed virtual infrastructure frameworks do not address it properly. In the HIPerNET framework, we propose to combine low-level machine virtualization with network virtualization and advance bandwidth reservation through service overlays based on full router virtualization to obtain the isolation of network performance.

2.2 HIPerNET and Virtual Private Execution Infrastructures concept

2.2.1 Design principles of the HIPerNET framework

The HIPerNET framework combines system and networking virtualization technologies with crypto-based-security, bandwidth sharing and advance reservation mechanisms to offer dynamic networking and computing infrastructures as services. This framework is transparent to all types of upper layers: upper layer protocols (e.g., TCP, UDP), APIs (e.g., sockets), middleware (e.g., Globus [3], Diet [11]), applications, services and users. It helps end users with intensive-application deployment, execution and debugging, by providing an environment to run them on scalable, controlled, distributed and high-capacity platforms. HIPerNET is also mostly agnostic of lower layers and can be deployed over IP as well as lambda networks.

The HIPerNET framework is responsible for the creation, management and control of dynamic entities called VPXIs, as defined in the next section, providing a generalized infrastructure service. It supervises VPXIs during their whole lifetime. The key principle is to have operations realized automatically. For this, HIPerNET defines and activates mechanisms to automatically control the VPXIs as well as the status of the underlying exposed substrate called HIPerSpace. HIPerNET integrates classical FACPS (fault-tolerance, accounting, configuration, performance and security) management functions to these dematerialized entities. In the context of this paper, we focus on two critical aspects of the virtualisation context: network configuration and performance isolation. Neither the security mechanisms—which are other key points of this framework [12]—relying on SPKI and the HIP protocol, nor the advance reservation algorithms or the migration mechanisms for fault-tolerance, are described in this paper.

Figure 1 illustrates two virtual execution infrastructures (VPXI A and VPXI B) allocated on a global exposed infrastructure (HIPerSpace) using the HIPerNET framework[1]. At the lower level, the HIPerNET framework accesses and controls a part of the physical infrastructure which is virtualized and called the HIPerSpace. Enrolled physical resources are then registered to the HIPerNET registrar and can be allocated to VPXIs. Once the resources have been exposed, HIPerNET gets full control over it. This set of exposed virtualized resources composes the substrate hosting the VPXIs. At run-time, the HIPerNET manager communicates with the nodes of the HIPerSpace to deploy virtual nodes, monitor their status and configure control tools to supervise the resource usage.

In this fully-virtualized scenario, HIPerNET interacts with multiple resource providers to plan, monitor and control them. Functions such as fault management, load balancing, bandwidth management and performance control are handled taking both network- and resource-virtualization techniques into account.

The current HIPerNET software implements virtual links at layer 3 of the OSI model (using IPsec and the HIP protocol [14]) enabling multiple virtual overlay networks to cohabit on a shared communication infrastructure. An overlay network has an ideal vantage point to monitor and control the underlying physical network and the applications running on the VMs. All network overlays and virtual end hosts

[1]*The HIPerNET software is under development within the ANR HIPCAL project [13]*

DOI: 10.1002/nem

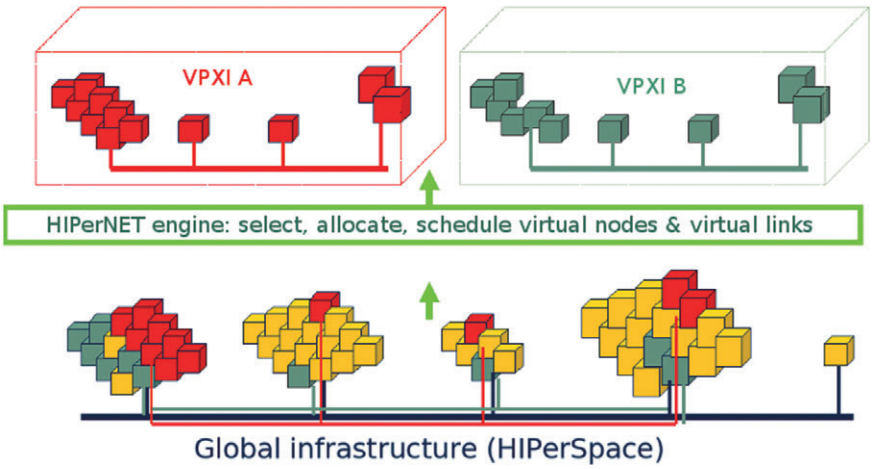

Figure 1. Example of a VPXI allocation on a distributed and virtualized HIPerSpace

are kept isolated both at the network and OS level to ensure security, performance control and adaptability. We also study the implementation of these virtual links at lower level (layer 2 or layer 1) within the CARRIOCAS project[10], [15].

2.2.2 Virtual Private Execution Infrastructure concept

We define the Virtual Private eXecution Infrastructure (VPXI) concept as the HIPerNET's management unit. VPXIs are an organized aggregation of virtual computing resources interconnected by a virtual private overlay network for a defined time period. Ideally, any user of a VPXI has the illusion that he is using his own dedicated system, while he really is using multiple systems, part of the global distributed infrastructure. The resulting virtual instances are kept isolated from each others and the members of a VPXI have a consistent view of a single private TCP/IP overlay, independently from the underlying physical topology. A VPXI can span multiple networks belonging to disparate administrative domains. In virtual infrastructures, a user can join from any location and use the same applications he was using on the Internet or its intranet.

Each VPXI is specified according to the user requirements. The specification is interpreted, allocated and configured using basic entities, which are building blocks composing the virtual execution infrastructures, as VXNodes: virtual end hosts, VXRouters: virtual routers, and VXLinks: virtual links.

VXNodes: Virtual end hosts. VXNodes or virtual end hosts are personalized virtual machines that are allocated in physical hosts, and managed by HIPerNET to make sure they share the physical resources among them in a confined, secured and isolated way. Developed in accordance with the specifications of users and applications, each virtual end host can be parameterized (e.g., CPU, RAM memory, storage) and configured (e.g., operating system, communication tools) respecting the key set of parameters informed during their specification. Otherwise, end hosts can be organized individually or in groups, recursively. Basically, a VPXI represents a complex infrastructure, composed by individual end hosts that interact with groups (clusters or grids). Groups allow HIPerNET to manage a set of virtual end hosts with the same characteristics all together, thus avoid unnecessary complex topologies and configuration scenarios.

VXRouters: Virtual routers. Within the VPXI design, virtual routers, which we call VXRouters, are fully-personalizable components of the VPXIs. For example, these VXRouters, instantiated within physical Linux routers, will run inside isolated virtual machine instances. In our approach, all traditional network planes (data, control and management) are virtualized. Therefore, users can use any protocol and control

DOI: 10.1002/nem

mechanism on their allocated virtual router, and within the virtual interconnection network. HIPerNET implements a default routing mechanism on the virtual routers, but users can reconfigure it. They can deploy customized routing protocols, and configure the packet-queuing disciplines, packet filtering and monitoring mechanisms they want. HIPerNET only manages the physical substrate and the resources attribution to each VXRouter. Also, VXRouters represent strategic points of the network for rate control as they concentrate aggregated VPXI traffic. By limiting the rate and policing the traffic of the different VXRouters, the traffic of VPXIs can be controlled and the user is provided with a fully isolated execution infrastructure. The control is implemented in the substrate, hence its implementation and management is transparent to the user. The advantage of having controlled environments is twofold: users have strong guarantees, while the network provider can better exploit the network by sharing it efficiently but differently between users.

VXLinks: Virtual links. Virtual execution infrastructures can be composed by several VXNodes, groups of them, and VXRouters. All these components are interconnected using VXLinks or virtual links. Each VXLink is a temporary path allocated among multiple physical channels. As in a real infrastructure, each link can be parameterized with different metrics according to the applications' requirements. Metrics such as bandwidth, latency and direction can be valuated for each virtual link. Using the source-destination definition, it is possible to organize the model of the virtual infrastructure, defining and asking for specific configurations in critical paths, that can interfere with the execution of applications.

HIPerNET comprises a module to specify virtual infrastructures. This module relies on the VXDL [16] language and provides users with a large flexibility in VPXI design as illustrated in the next section.

3. VXDL: SPECIFYING AND MODELING VIRTUAL PRIVATE EXECUTION INFRASTRUCTURES

3.1 Needs for a specific language

Users need to define the virtual private execution infrastructures they request according to the applications' requirements. This process requires a specific language to describe virtual resources and networks. Existing languages are used in different areas of networks or IT resources management. The Common Information Model Specification (CIM) [17] provides a common set of objects and relationship among them. The CIM architecture is based on the UML concept and provides language CQL (CIM Query Language) to select sets of properties from CIM-object instances. CQL is a subset of SQL-92 with some extensions specific to CIM. In this case, CQL allows queries asking for a set of resources with certain configurations, but does not have parameters to interact with the allocation system, for example informing the basic location of a component. An other common example is the Network Description Language (NDL) [18]. NDL is a collection of schemas (topology, layer, capability, domain and physical) used to represent a network infrastructure at different levels. This language is guided by the Resource Description Framework (RDF) [19], a general-purpose language for representing information on the Web. Basically, RDF (and NDL) explores a graph data model composed by a set of RDF triples (a subject, an object and a predicate (property)). The triple concept is difficult to adapt to recursive description, like groups or aggregates found in virtual infrastructures. A descriptive language dedicated to virtual infrastructures description must be more adaptive than conventional solutions and needs to combine the space and temporal aspects of virtual infrastructures. During the specification process, new challenges coming from virtualization techniques have to complement the management, configuration and execution features of classical languages. We identify some desirable aspects that must be explored for descriptive languages: a) recursive representation of individual resources and groups; b) each resource must be characterized using an extensible list of descriptive parameters to represent the necessary

DOI: 10.1002/nem

configuration—e.g., RAM memory, CPU speed, storage capability; c) considering the configuration and deployment is also necessary to describe the software (e.g., operating systems) and tools (e.g., communication and programming tools) that must be used in VPXI; d) each component (individually or in group) should be characterized as its elementary function, for example to describe a set of computing nodes or storage nodes; e) it must be possible to express the composition of virtual network topology, detailing each virtual link. On the other hand, parameters should enable an abstract description, for example, informing a virtual link description that must be used to interconnect a group of resources; f) the executing timeline of the infrastructure (this parameter should help in resources reservation and co-scheduling); g) security attributes for each resource element (access control, confidentiality level); h) commercial attributes for each resource element (maximum cost); and i) temporal attributes for each resource element (time window for provisioning).

Within a given time slot, a VPXI can be formally represented as a graph—G(V, E)—where V is a set of vertices and E represents a set of edges. A vertex is in charge of active data processing functions and an edge is in charge of moving data between vertices. Figure 2 illustrates this concept, representing a virtual infrastructure composed by the aggregation of virtual machines interconnected via virtual channels. It shows two VXRouters (vertices rvA and rvB) which are used to interconnect and perform the network control functions among the other virtual resources (vertices rv 1 to 8). These virtual routers can independently forward the traffic of the different virtual infrastructures sharing the same physical network. Each edge represents a virtual link used to interconnect a pair of virtual resources, which contains different configurations, as lv 1 and lv 2. Note that at the next time slot, the configuration may be different.

To specify a virtual execution infrastructure we introduce VXDL (Virtual Resources and Interconnection Networks Description Language), a language that besides allowing end-resource description lets users describe the desired virtual network topology (using the same grammar), including virtual routers and timeline representation. Implemented with the XML standard, VXDL helps users and applications to create or change VPXI specifications. The VXDL grammar is divided into *Virtual Resources description, Virtual Network Topology description* and *Virtual Timeline description* [16]. A new key aspect of this language is that all these basic descriptions are optional: it is possible to specify a simple communication infrastructure, or a simple end-resource aggregate.

3.2 Virtual resources description

The key point explored in this part of the VXDL grammar is enabling users and applications to describe in a simple and objective way all the necessary end hosts, and groups of them. The parameters proposed

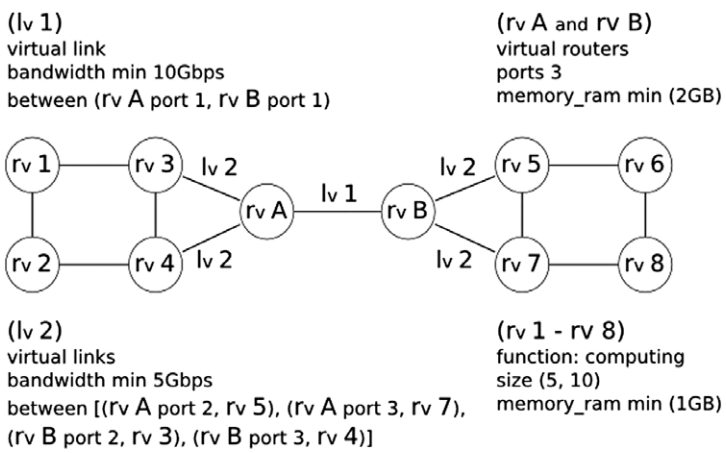

Figure 2. Example of a VPXI composition using graph notation

DOI: 10.1002/nem

in VXDL are directly related to virtual infrastructures, and besides allowing basic resource parameterization (e.g., maximum and minimum values acceptable to RAM memory and CPU frequency), VXDL lets users directly interact with management frameworks (such as HIPerNET). For this reason, VXDL suggests the definition of parameters such as anchor, number of virtual machines allocated per physical host, and key components.

The anchor parameters act in the physical allocation of a VPXI. We know that in a virtualized substrate the VPXI can be allocated anywhere, but sometimes it is necessary that a virtual end host (or group) must be positioned in a physical location (e.g., a site or a machine—URL, IP) in accordance with a basic location. Yet, analyzing a virtualized substrate, multiple virtual machines can be allocated in the same physical host, sharing the real resources among them. VXDL enables the definition of a maximum number of virtual machines that must be allocated in a physical host, enabling users to interact directly with the allocation algorithm[2]. The key parameter acts in the identification of the most important components of the VPXI. This information will help in the allocation, informing that the component must be allocated in advance.

3.3 Virtual network topology description

VXDL improves the network topology description in two key points: I) enabling the specification together with other components, using the same grammar; II) applying the concept of link organization, which permits a simple and abstract description. Links can define connections between end hosts, between end hosts and groups, inside groups, between groups and VXRouters, and between VXRouters. Using this concept users can model their VPXIs in accordance with their applications. Figure 3 presents a possible VPXI composed by two groups (clusters), each with 8 nodes interconnected using 2 VXRouters.

In VXDL grammar, the definition of source-destination pairs for each link is proposed. A single link can be applied to different pairs, simplifying the specification of complex infrastructures. For example, links used to interconnect a homogeneous group, such as a cluster, can all be defined in a same link description. Each link can receive a parameterization involving latency, bandwidth and direction. To latency and bandwidth, the maximum and minimum values are applied.

3.4 Virtual timeline description

Any VPXI can be permanent, semi-permanent or temporary. The VPXIs are allocated for defined lifetime in time slots. Time slots are configurable parameters and are specific to the HIPerSpace. Often the VPXI components are not used simultaneously or all along the VPXI's lifetime. Thus, the specification of an internal timeline for each VPXI can help the middleware in the allocation, scheduling and provisioning processes. Periods can be defined in function of their usages, as data transfer or computation, delimited by temporal marks. A period can start after the end of another period or after an event.

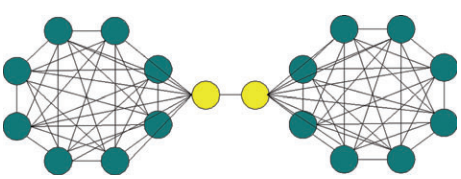

Figure 3. A graph representing a network topology

[2]*More information about VXDL is provided on http://www.ens-lyon.fr/LIP/RESO/Software/vxdl*

DOI: 10.1002/nem

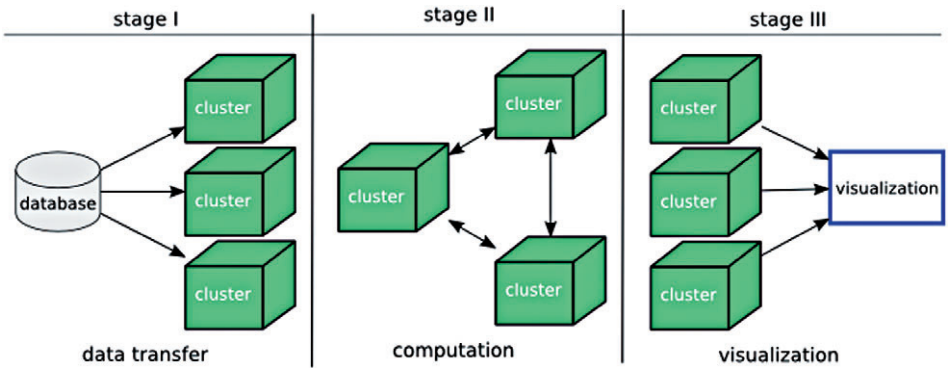

Figure 4. Example of a virtual timeline definition

Looking at figure 4 we can observe three execution stages of an application. This VPXI is composed by one data storage cluster, three computation clusters and one visualization cluster, all interconnected through virtual links. In stage I, it is necessary to perform a data transfer between the storage cluster and the computation clusters. During this transfer, a high-bandwidth configuration is required, which is not necessary in the others stages. As early as in stage II, the three computation clusters execute an application, possibly requesting links with a low latency in communication to the substrate framework. Finally, stage III asks for a high bandwidth again to perform the data transfer and results visualization. In this example, the periods between stages I and II or stages II and III can be defined in a relative way: after the data transfer, in stage II, computation starts and the low latency on the intra-cluster links is implemented. VXDL explores the relative definitions using common parameters as start, after and until.

4. DEPLOYMENT AND PROVISIONING VPXIs

In a typical scenario, HIPerNET receives a user's VPXI request, specified in VXDL. The software interprets it, then checks its feasibility and potentially negotiates security, performance, planning and price with the customer or its representative. When an agreement has been found, a VPXI instance is allocated and scheduled as proposed in [20]. The VPXI is then provisionned according to the schedule. Finally, the deployment of a VPXI consists in instantiating and configuring the specified virtual entities (VXNodes, VXRouters and VXLinks) on top of the physical substrate. First, all the virtual nodes (end hosts and routers) are deployed, then the virtual links and routes are set up. Routes are configured individually in all the VXNodes and VXRouters. During the VPXI's lifetime, its allocation may change as described in section 3.4. In this case, if the topology changes, new VXLinks are deployed if necessary and the routes are reconfigured in all the VXNodes and VXRouters to reconnect the topology. Below, we describe the mechanisms that are deployed to virtualize the network resources.

4.1 VPXI network resource provisioning

Virtual infrastructures share a physical substrate network. To ensure VPXI isolation, the resource sharing between different virtual infrastructures needs to be finely controlled. To achieve this, a set of control mechanisms residing inside the physical substrate nodes are set up. They are in charge of supervising, at runtime, how the physical capacity is allocated to the virtual nodes and links. After the VPXI allocation

DOI: 10.1002/nem

and scheduling process, virtual capacities are assigned to each virtual node and link. Then, these capacities are reserved and provisionned in the physical substrate. During the deployment phase, the control tools are invoked at each involved substrate node, in order to enforce allocations during the VPXI lifetime. At runtime, the control tool's actions can be reconfigured when the VPXI's requirements change. This happens for example when the execution moves from one stage to the next. At that moment, HIPerNET updates the configuration of the deployment according to the allocation corresponding to the new stage. Thus, it reconfigures the control tools, in order to change the resource attribution. The control tools have three main features: they manage the bandwidth, CPU and memory sharing among the virtual domains, so that they offer the user provisionned VPXIs according to his requirements. During the negotiation, a VPXI user can specify a desired bandwidth per virtual link, associated to one of the following services:

- *Guaranteed minimum:* the minimum rate that has to be available on the virtual link at any moment;
- *Allowed maximum:* the maximum capacity the virtual link must allow at any moment;
- *Static reservation:* in this case, the guaranteed minimum is equal to the allowed maximum.

The user can use one of these services to specify its application's requirements. A user who wants to execute, for example, a distributed application communicating with MPI will specify a guaranteed minimum rate which can not be decreased during negotiation. If the user specifies an allowed maximum, the negotiation will result in a bandwidth equal or below this maximum. This link is shared in a best-effort way with other traffic. Specifying a static rate gives the user the impression that he works in a dedicated physical network whose links have the specified capacity. He will be able to obtain the specified bandwidth at any moment but can never exceed it. This kind of service allows no negotiation and could for example help with conducting reproducible experiments where link availability should not vary.

According to these user specifications and the free capacity on the physical substrate, a static bandwidth reservation is made. Then, HIPerNET's control mechanism allocates the maximum bandwidth for each virtual link in order to guarantee the negociated bandwidth. The configuration takes place on both ends of the virtual link configuring the physical interfaces.

Let's consider two users of two VPXIs who specified different rates for the different virtual links. The user of VPXI 1 desires a bandwidth of 100 Mb/s on virtual link A (VL A) and a bandwidth of 200 Mb/s on VL B connected to VXRouter (VXR) 1. The user of VPXI 2 wants both virtual links connected to VXRouter 2, VL C and VL D, to allow 400 Mb/s. Figure 5 shows the specified virtual network configurations of VPXI 1 and VPXI 2. HIPerNET configures the virtual link rates to these values during the deployment of VPXI 1 and VPXI 2.

Users can also specify a maximum percentage of the physical CPU a virtual node of a VPXI should be allowed to use. This quantity of CPU is allocated in the available CPU of the physical machine. The

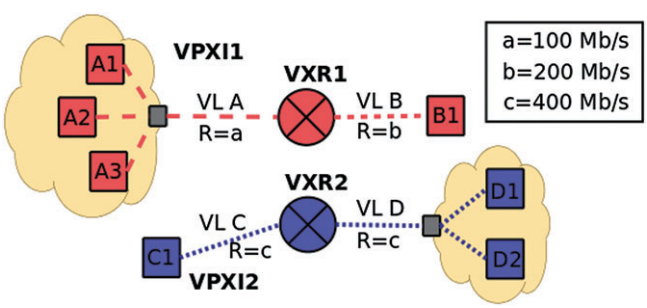

Figure 5. User bandwidth specification for two VPXIs

DOI: 10.1002/nem

available quantity of CPU corresponds to the CPU of the physical machine minus the part of the CPU in use for the machine's host system, or domain 0 (dom0) in the Xen terminology:

$$CPU_{availabel} = CPU_{machine} - CPU_{dom0}$$

Dom0 consumes a small part of CPU time for the control mechanisms and a considerable part for networking as it is in charge of forwarding the packets of all the virtual machines, or domains U (domUs):

$$CPU_{dom0} = CPU_{control} + CPU_{networking}$$

and

$$CPU_{control} \ll CPU_{networking}$$

So CPUdom0 can be established according to the sum of the maximum rates specified by all the domUs hosted on the considered physical machine. In Section 5.2, this relationship is evaluated on virtual routers.

The next section gives more details on the VXRouters.

4.2 Implementation of VXRouters

In the current HIPerNET implementation, the VXRouters consist in software routers implemented inside virtual machines. These VXRouters are hosted by network-aware servers, commodity servers with the required hardware to support high-performance networking (several network interface cards, several CPUs and a huge amount of memory) used as Linux routers. Current commercial virtual or logical routers do not support the full virtualization of the router data planes. The performance impact of the data-plane virtualization is indeed an issue. However, recent analysis have demonstrated that modern virtualization techniques are improving rapidly, making our VXRouter approach interesting and promising [21]. Figure 6 shows an example of a physical router hosting two VXRouters. These virtual routers have to share the physical resources of the machine and additional processing is necessary.

Customized routing and traffic-engineering functions can be set up on the VXRouters. A VPXI user can, for example, choose to adapt the routing in order to satisfy specific Quality of Service requirements, as illustrated in Figure 7.

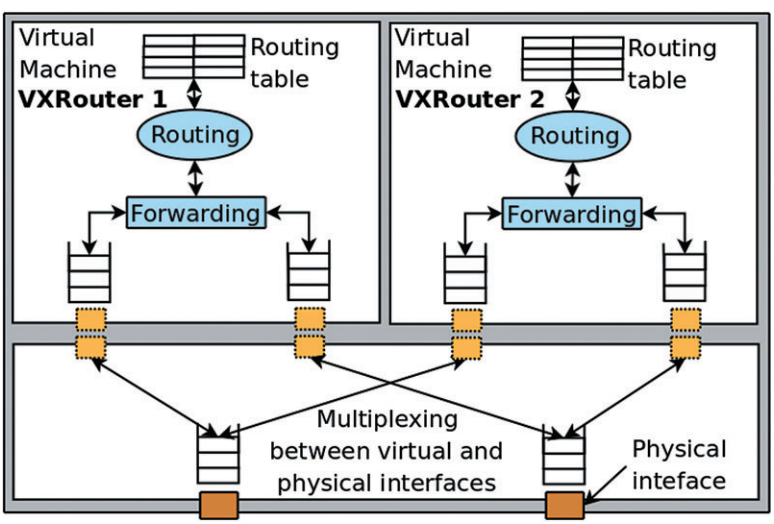

Figure 6. Model of a physical router hosting VXRouters

DOI: 10.1002/nem

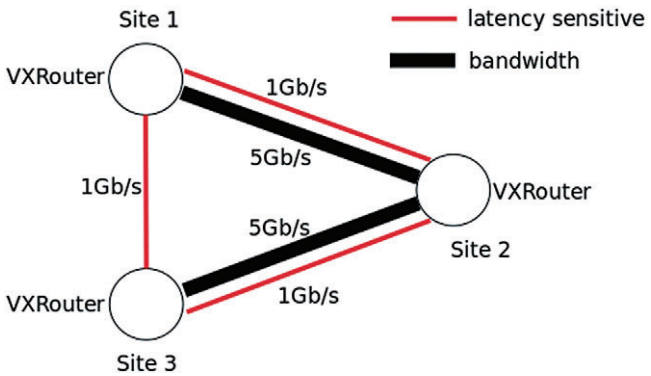

Figure 7. Example of bandwidth allocation for latency-sensitive and high-bandwidth flows

In this example, latency-sensitive traffic and bandwidth-aware traffic are routed on different paths. 1 Gb/s links are allocated between each one of the three VXRouters to transmit latency sensitive traffic. All high-throughput traffic, which can be pipelined, is redirected over Site 2. The advantage of this routing scheme is that the user needs to provision only 2 links with 5 Gb/s instead of 3. As presented in [22], this is an example of efficient channel provisioning combining routing and traffic engineering.

5. EXPERIMENTS

This section presents experimental results of the current implementation of network virtualisation in HIPerNET. Rate control for bandwidth provisioning, performance isolation and the CPU cost of bandwidth are investigated. To instantiate virtual end hosts and virtual routers, we chose to use the Xen [6] technology because of the offered flexibility: distinct operating systems with distinct kernels can run in each virtual machine and the user is provided with full configurability to implement individual virtual networks. All the experiments are executed within the Grid'5000 [23] platform, using Xen 3.2 and IBM Opteron servers with one 1 Gb/s physical interface. The machines have two CPUs (one core each).

5.1 Rate control and performance isolation experiment

The goal of this experiment is to evaluate the isolation between VXRouters when they are mapped on the same physical router.

5.1.1 Experimental setup

Two VXRouters called VXR1 and VXR2 share one physical machine to forward the traffic of two VPXIs (VPXI 1 and VPXI 2). Each virtual router has two virtual interfaces connected to two virtual links as represented on Figure 8. These links are configured to allow a maximum bandwidth R of respectively $R_{VLA} = R_{VLB} = 150$ M b/s and $R_{VLC} = R_{VLD} = 300$ Mb/s. Figure 8 represents this setup.

Three cases of user traffic can be distinguished by

1. *in-profile* traffic: the sum of the traffic on virtual link X is smaller than the allowed maximum rate $R_{VL\,X}$;
2. *limit* traffic: the sum of the traffic on virtual link X is equal to the allowed maximum rate;
3. *out-of-profile* traffic: the sum of the traffic on virtual link X exceeds the allowed maximum rate.

This experiment aims at determining if the traffic-control techniques limit the traffic to the desired rate and if isolation is guaranteed. Isolation means that limit or out-of-profile traffic should have no impact

DOI: 10.1002/nem

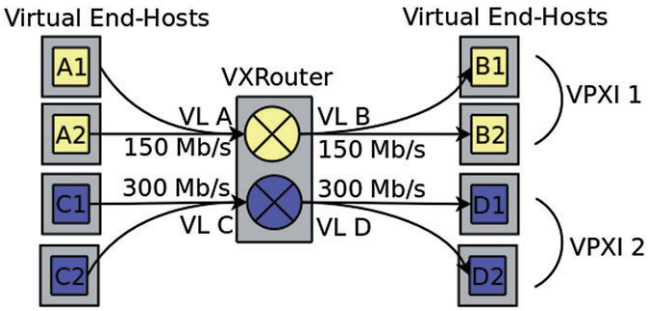

Figure 8. Experimental setup with 2 VPXIs with different bandwidth requirements

	VPXI 1	VPXI 2
Case 1	In profile	In profile
Case 2	Limit	Limit
Case 3	In profile	Limit
Case 4	In profile	Out of profile

Table 1. User traffic for different test cases

on the available bandwidth on the physical link, and thus none on the available bandwidth of concurrent virtual links.

Table 1 lists four testcases to validate our implementation. In this experiment, we suppose a congestion factor (CF) of 0.8 for in-profile traffic, which means that traffic is sent at 80% of the maximum allowed speed. Limit traffic has a CF of 1 and for out-of-profile traffic, a CF of 1.2 is chosen.

One flow is sent from virtual end hosts Ai and Ci to the corresponding virtual end hosts Bi and Di, respectively. The flows are sent in the different cases so that their sum on each VXRouter corresponds to the profile specified in Table 1.

5.1.2 Results

The results of these experiments show that the desired rate on each virtual link can be obtained with our configuration and that out-of-profile traffic does not impact other traffic sharing the same physical link.

In case 1, where all the traffic is in profile, requiring less bandwidth than allowed, each flow gets its desired bandwidth and no packet loss is detected with UDP. For case 2, the flows try to get 100% of the maximum available bandwidth. This causes some sporadic packet losses which cause some instantaneous throughput decreases in TCP as illustrated in Figure 9. The overall throughput still reaches the allowed maximum value.

Figures 10 and 11 show respectively the results with TCP and UDP flows in testcase 4 where the two flows of VPXI 1 are in profile, their sum 30 + 90 = 120 Mb/s does not exceed the maximum allowed value of 150 Mb/s. With TCP and UDP, both flows attempt their desired rate and with UDP, no loss is detected. The two flows of VPXI 2 try to exceed the allowed value of 300 Mb/s by sending at 120 + 240 = 360 M b/s. As a result, they see some of their packets dropped at the incoming interface of the physical router. TCP reacts in a sensitive way to this packet drop. It tries to share the available bandwidth by the two flows. So flow 1 gets generally the 120 M b/s as it is less than half of the available 300 Mb/s. Flow 2 requiring 240 Mb/s varies a lot to get also an average throughput of about half of the 300 Mb/s.

With UDP (Figure 11), the packet drop of the flows of VPXI 2 causes a regular decrease of the bandwidth. Flow 1 looses an average 13% of its packets and flow 2 an average of 18% of its packets. The

DOI: 10.1002/nem

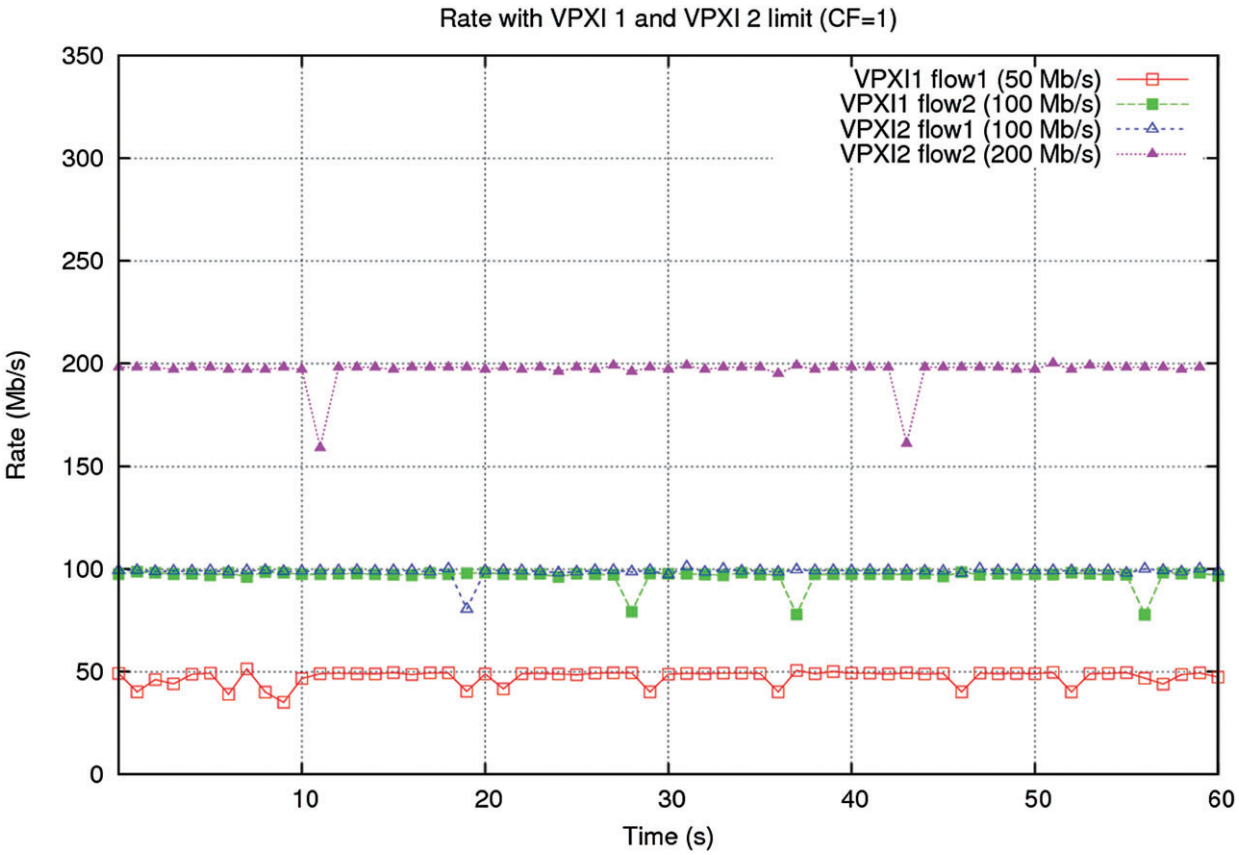

Figure 9. Test case 2: TCP Rate with VPXI 1 and 2 being at the limit rate (with congestion factor(CF) of 1)

average of these two values is slightly smaller than the percentage of exceeding packets (1 – (1/1.2) = 16.6%) implied by the congestion factor of 1.2.

These results show that only flows which try to exceed the fixed limit bandwidth are penalized in this configuration. *In-profile* and even *limit* flows are not impacted. In this way, individual rate sharing is efficient in this model and isolation is guaranteed.

5.2 CPU cost of bandwidth

To find out how much CPU is needed to forward flows in domain U, making use of data-plane virtualization, and how this value depends on the forwarding rate, this experiment measures the CPU usage of Xen during forwarding on VXRouters with different rates. This experiment is executed with 1, 2 and 4 concurrent VXRouters on a single physical router, each one forwarding a single UDP flow with different rates. The rate of each flow is incremented every 60 seconds so that the aggregate rate on the set of virtual routers increments by 10 Mb/s. During each interval, the CPU use of the physical router is measured with Xen's *xentop* tool. Figure 12 shows the CPU utilization of dom0 of the physical router in the different cases. It increases with the increasing aggregate forwarding rate in the VXRouters. Surprisingly, this overhead does not significantly vary with different numbers of VXRouters. It is even almost identical when 2, 4 or 8 VXRouters are forwarding simultaneously flows. The corresponding CPU utilization of a domU during the same experiments is represented on Figure 13. The overhead increases with the

DOI: 10.1002/nem

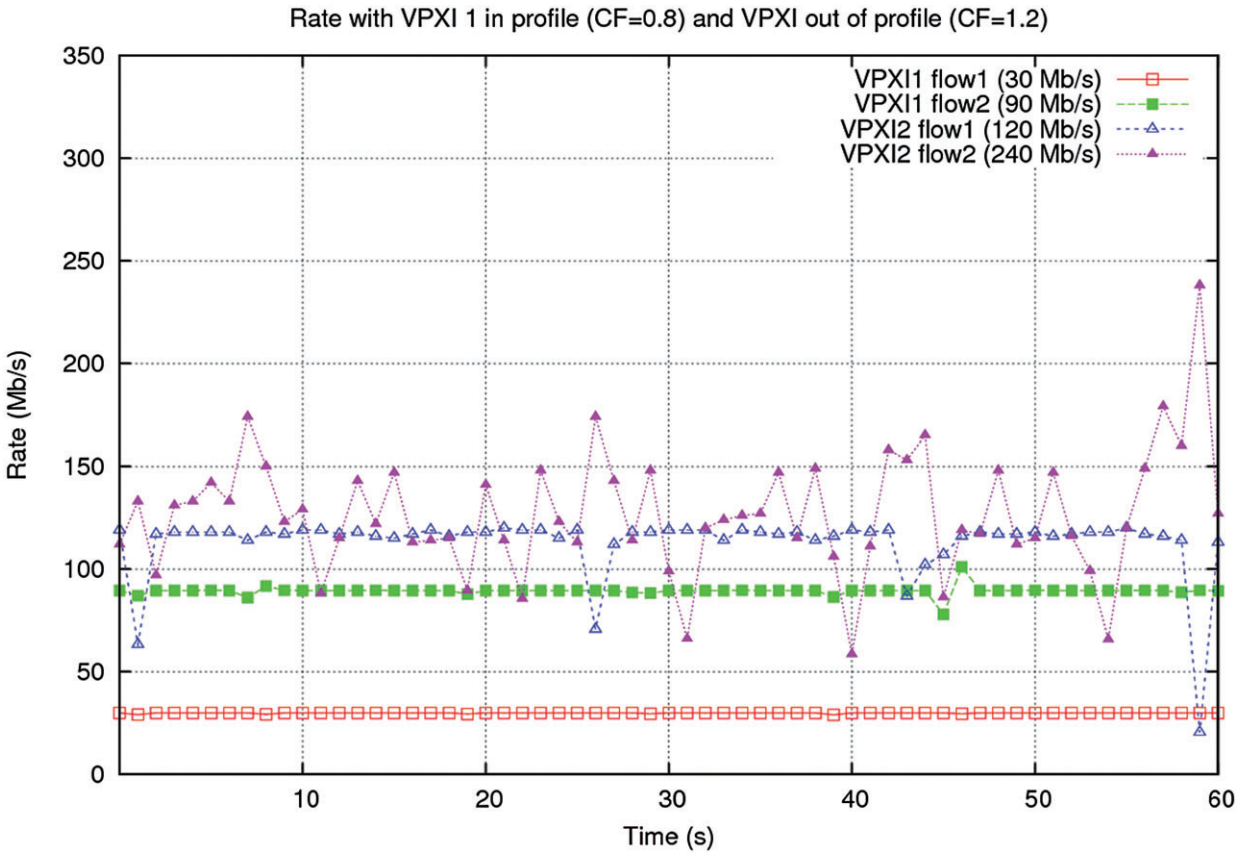

Figure 10. Test case 4: TCP Rate with VPXI 1 being in profile and VPXI 2 out of profile (with a congestion factor(CF) of 1.2)

forwarding rate, but in the case of dom0, it does not increase with the number of concurrent VXRouters. Only the cost of a single VXRouter represents an exception. This means that the CPU cost of a physical router hosting VXRouters depends strictly on the aggregate forwarding rate and not on the number of VXRouters it hosts.

6. RELATED WORK

Infrastructure virtualization is a hot topic studied in several projects [24,25,26]. In [24], the authors propose VINI, a virtual network infrastructure that allows several virtual networks to share a single physical infrastructure, similarly to HIPerNET. Researchers can run experiments in virtual network slices with XORP routing software, running inside UML instances. They can configure it and choose a protocol among the available ones. HIPerNET pushes this facility a step further adding data-plane virtualization and allowing users to chose the operating system and install any routing software. This also provides full isolation between virtual nodes and controlled resource sharing. VINI has dedicated IP address blocks, HIPerNET interposes a HIP-layer between the network and the application levels, that provides each resource with a unique identifier (HIT).

Trellis [27] is a network-hosting platform deployed on the VINI facility. Like HIPerNET, it is a virtual-network substrate that can run on commodity hardware. It is implemented using VServer's [28]

DOI: 10.1002/nem

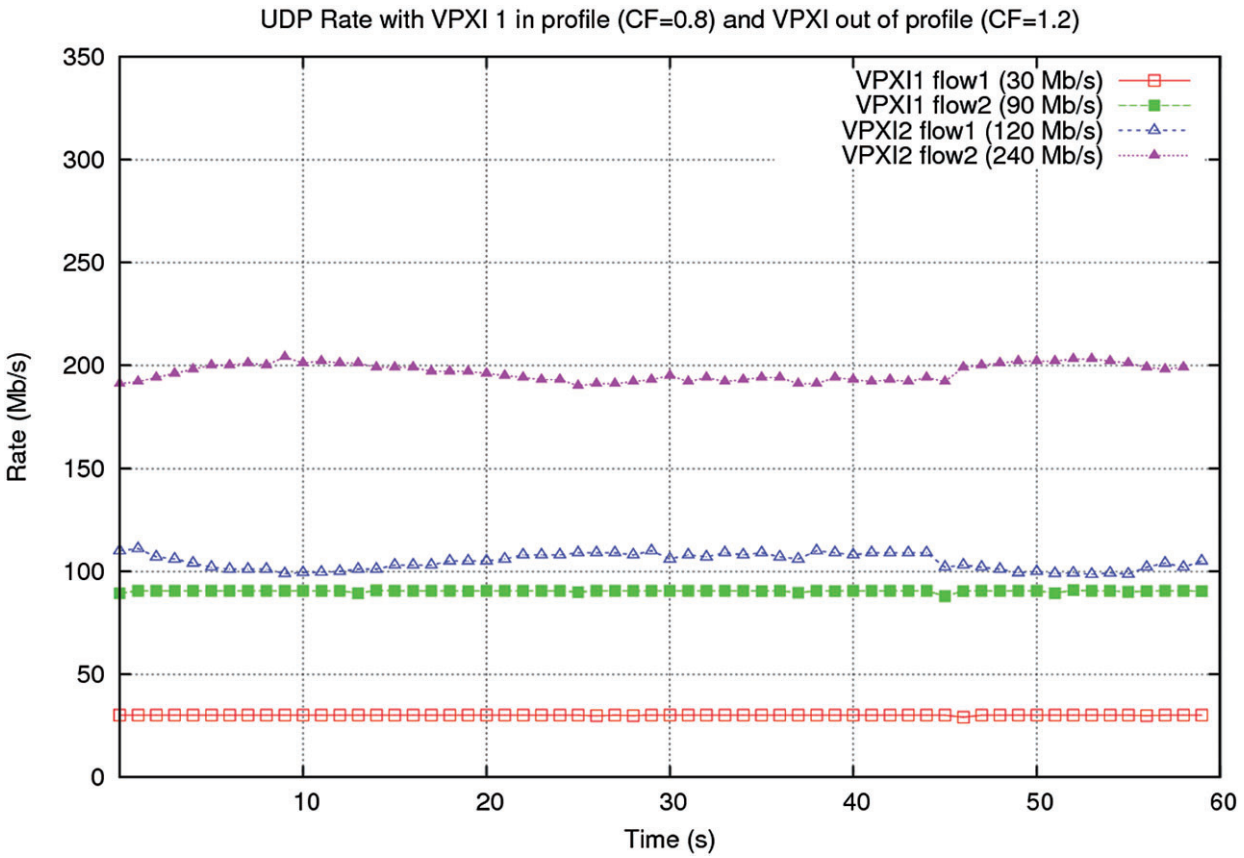

Figure 11. Test case 4: UDP Rate with VPXI 1 being in profile and VPXI 2 out of profile (with a congestion factor(CF) of 1.2)

container-based virtualization because of its networking performance. For the reasons described before, we use the Xen hypervisor which has become promising in terms of its growing network performance in its recent versions.

VINI and Trellis are evaluated on PlanetLab [29] which provides users with slices of virtual resources distributed among its overlay infrastructure. Those virtual resources are VServers providing generally end-host functionalities. In HIPerNET, we chose to integrate the resource 'virtual router' too, allowing users to custom the routing. HIPerNET is evaluated in a confined large-scale experimental facility.

In the GENI design [25] the users are provided with slices like in VINI, but these slices are composed of resources which can be either virtual machines or just partitions of physical resources. GENI's goal is to provide users with multiple shareable types of resources. The HIPerNET approach is compatible with GENI. Our goal in HIPCAL is to prove the concept by implementing it in a real testbed. The presented early results are promising. The idea to decouple services from the infrastructure with virtualization is also described in CABO [26]. CABO's argument is to give Internet service providers end-to-end control while using the physical equipments of different physical-infrastructure providers. HIPerNET combines network virtualization with end-host virtualization to provide users with virtual computing infrastructures, interconnected by an end-to-end controlable virtual network.

The main difference between these projects and HIPerNET is that HIPerNET provides users with full configurability. HIPerNET gives users root access to the virtual nodes. This means that users can deploy

DOI: 10.1002/nem

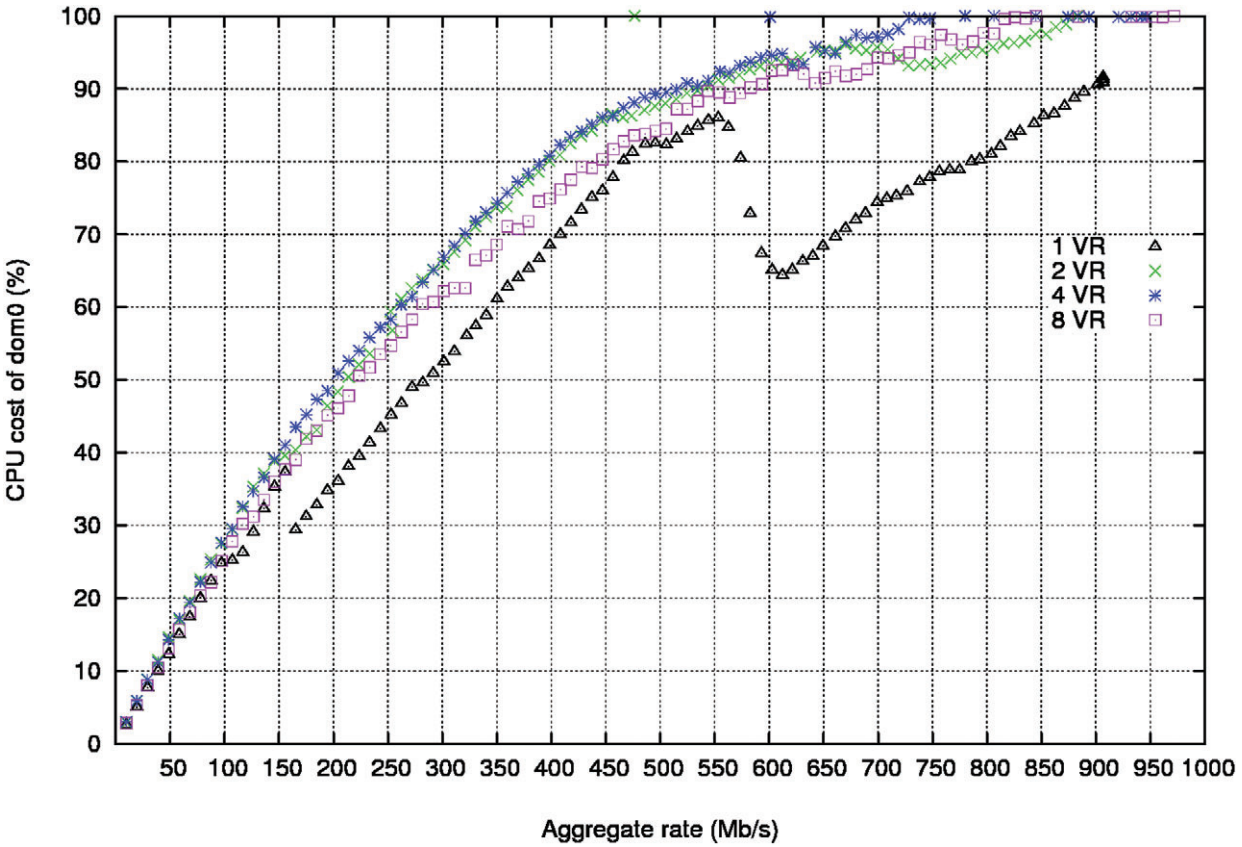

Figure 12. Dom0 CPU use with 1, 2, 4 or 8 concurrent VXRouters forwarding UDP flows

any operating system they choose on the node, and so have full configuration ability. It is then always possible to restrict configurability features for unexperimented users.

Resource sharing in terms of bandwidth in virtual networks has been treated by DaVinci [30], a design and mathematical model of dynamic adaptation of virtual networks to their performance objectives. While DaVinci's design enables optimal physical-link utilization through monitoring, HIPerNET's implementation focuses on giving strong performance guarantees and isolation to users, avoiding interferences if a virtual-network user tries to exceed his allocated rate.

Previous research on virtual routers has concluded that they have lower performance with Xen's data-plane virtualization than with control-plane virtualization only [31]. But as HIPerNET focuses on full configurability including OS choice, Xen's data-plane virtualization meets its requirements. In particular, Xen's data-plane virtualization performance has already been growing with the successive versions, now achieving Linux-like throughput, at least for big packets [21]. Moreover, major promising network improvements are expected in the next version [32].

Current hardware solutions like Juniper's TX Matrix Plus routers allow running up to 16 routing instances. By virtualizing the control-plane, they intend to offer a gain in routing capacity and reduce power consumption, a goal they share with server consolidation. Even though these routers do have very good performance, they are not as easily deployable at a very large scale and for experimentation purpose as VXRouters. They do not allow the same configuration facility. An interesting approach using virtual routers is the ability to migrate them over the substrate, to meet the resource requirements at any moment and optimize the overall resource utilization. This concept is described in [33]. In combination with the

DOI: 10.1002/nem

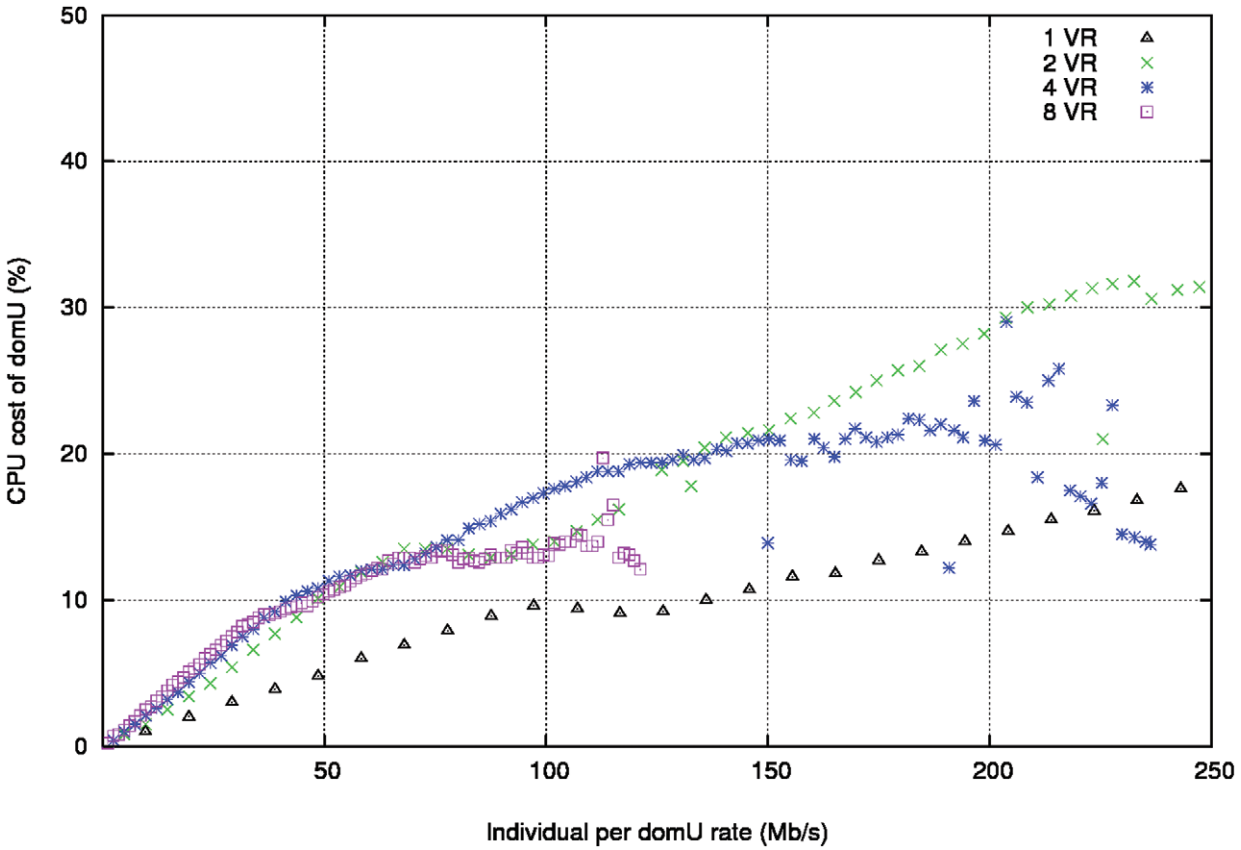

Figure 13. DomU CPU use with 1, 2, 4 or 8 concurrent VXRouters forwarding UDP flows

HIPerNET framework, this type of dynamic allocation would allow optimal resource utilization in accordance with the user-defined specification and is also interesting for fault tolerancy. In addition to the migration, resource utilization on the links between the virtual routers can also be optimized by allocating a virtual path to multiple physical links as suggested in [34]. This allows allocating virtual links, which require more capacity than is free on a single physical link. These extensions will be studied for integration in HIPerNET.

Existing commercial products, including Amazon's Elastic Compute Clouds (EC2)[3], Enomaly's Elastic Computing Platform (ECP)[4] or GOGRID[5], like HIPerNET, allow users to reserve a set of resources, choose an OS and customize it. Others are less configurable, like 3tera's AppLogic[6], which has no OS choice and b-hive[7], which is data-center oriented. All these clouds middleware are suitable to perform computation or storage and are not intended to host virtual routers, as no special physical routing machines exist to carry the virtual routers, these would be hosted on an end host like in overlay networks. Moreover these Cloud Computing solutions do not intend to control the network or the communication performance as HIPerNET does.

[3]*http://aws.amazon.com/ec2*
[4]*http://www.enomaly.com*
[5]*http://www.gogrid.com*
[6]*http://www.3tera.com*
[7]*http://www.bhive.net*

DOI: 10.1002/nem

7. CONCLUSION

The convergence of communication and computing on the Internet encourages a Networking and Computing Infrastructure as a Service (IaaS) approach. In this context, this paper put in perspective the concept of Virtual Private eXecution Infrastructure (VPXI), the VXDL langage and the HIPerNET software which is developed within the ANR HIPCAL project to create and supervise VPXIs multiplexed in time and space. HIPerNET allows users to submit VPXIs requests, and allows infrastructure owners to virtualize a physical substrate network to share it efficiently and dynamically. We overviewed the VXDL language—which conforms to the XML standard—that we have defined to enable users to dimension VPXIs precisely and to specify their attributes. Once specified, a VPXI request is submitted to the HIPerNET engine which processes it and allocates resources accordingly. HIPerNET is responsible for admission control, resource discovery and allocation, and VPXI management in terms of fault, accounting, performance, configuration and security. This paper concentrated on the network-resource configuration feature of the HIPerNET framework. HIPerNET integrates virtualized software routers, VXRouters, to give users the ability to fully customize networking and routing functions. A validation of the link-control services for managing and sharing bandwidth between the virtual networks has been presented, as well as an evaluation of the data-lane virtualization in terms of CPU overhead. The results showed that, thanks to the HIPerNET mecanisms, in profile and limit flows are not impacted by non-conforming traffic. In this way, we demonstrated that in our implementation of the HIPerNET model, individual rate control is efficient and isolation is guaranteed. We also showed that the CPU cost of a physical software router hosting VXRouters depends strictly on the aggregate forwarding rate and not on the number of VXRouters it hosts. We are currently deploying the HIPerNET framework integrating the virtual routers and virtual links control in Grid'5000's 10 Gb/s context for isolating VPXIs dedicated to distributed applications experiments. We are also investigating security, resiliency and performance aspects with real user applications.

ACKNOWLEDGEMENTS

This work has been funded by INRIA and the French ministry of Education and Research via the HIPCAL ANR grant and by the CARRIOCAS pôle System@tic grant. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS, RENATER and other contributing partners (see https://www.grid5000.fr). The authors would like to thank Augustin Ragon for his contribution as well as Olivier Mornard and Jean-Patrick Gelas for their help in the development and validation of the HIPerNet software. The authors would like to thank the anonymous reviewers for their fruitful comments and suggestions.

REFERENCES

1. RESERVOIR Seed Team. RESERVOIR: An ICT Infrastructure for Reliable and Effective Delivery of Services as Utilities, *Technical Report H-0262*, IBM Research Division, February 2008.
2. Evaluation of current network control and management plane for multi-domain network infrastructure, *FEDERICA Deliverable DJRA1.1*, July 2008.
3. Foster I, Kesselman C. Globus: A metacomputing infrastructure toolkit. International *Journal of Supercomputer Applications*, 1997; **11**: 115–128.
4. EGEE: Enabling Grids for E-sciencE. http://www.eu-egee.org [3 October 2009].
5. Popek GJ, Goldberg RP. Formal requirements for virtualizable third generation architectures. In *SOSP 73: Proceedings of the fourth ACM Symposium on Operating Systems Principles*, New York, 1973; p. 121.

DOI: 10.1002/nem

6. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. In *SOSP 03: Proceedings of the 19th ACM Symposium on Operating Systems Principles*, New York, 2003; 164–177.
7. Sugermann J, Venkitachalam G, Lim B-H. Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. *In Proc. 2001 Usenix Annual Technical Conference, pp. 1–14, Usenix Assoc., 2001.*
8. Duffield NG, Goyal P, Greenberg A, Mishra P, Ramakrishnan KK, van der Merive JE. A flexible model for resource management in virtual private networks. *SIGCOMM Computer Communication Review* 1999; **29**(4): 95–108.
9. Finlayson M, Harrison J, Sugarman R. *VPN Technologies: A Comparison*. Data Connection: London, February 2003.
10. Vicat-Blanc Primet P, Soudan S, Verchère D. Virtualizing and scheduling optical network resource for emerging IT services. *Journal of Optical Communications and Networking* 2009; **1**: A121–A132.
11. Caron E, Desprez F. DIET: A Scalable Toolbox to Build Network Enabled Servers on the Grid. In *Int. Journal of High Performance Computing Applications*, 2006; **20**(3): 335–352.
12. Vicat-Blanc Primet P, Gelas J-P, Mornard O, Koslovski G, Roca V, Giraud L, Montagnat J, Huu TT. A scalable security model for enabling dynamic virtual private execution infrastructures on the internet. In *IEEE International Conference on Cluster Computing and the Grid (CCGrid2009)*, Shanghai, May 2009.
13. HIPCAL project. http://www.ens-lyon.fr/LIP/RESO/Projects/HIPCAL/ProjetsHIPCAL.html.
14. Moskowitz R, Nikander P. Host Identity Protocol (HIP) architecture. IETF request for comments. *RFC 4423*, May 2006.
15. Calcul Réparti sur Réseau Internet Optique à Capacité Surmultiplée (CARRIOCAS), 2007. http://www.carriocas.org/
16. Koslovski G, Vicat-Blanc Primet P, Charão AS. VXDL: Virtual Resources and Interconnection Networks Description Language. In *GridNets 2008*, Oct. 2009.
17. Common Information Model (CIM) standards. http://www.dmtf.org/standards/cim/ [3 October 2009].
18. van der Ham J, Dijkstra F, Travostino F, Andree H, de Laat C. Using RDF to describe networks. *Future Generation Computer Systems* 2006; **22**: 862–867.
19. Resource Description Framework (RDF)/W3C semantic web activity. http://www.w3.org/RDF/ [3 October 2009].
20. Koslovski G, Truong Huu T, Montagnat J, Vicat-Blanc Primet P. Executing distributed applications on virtualized infrastructures specified with the VXDL language and managed by the HIPerNET framework. In *First International Conference on Cloud Computing (CLOUDCOMP 2009), (Munich, Germany)*, October 2009.
21. Anhalt F, Vicat-Blanc Primet P. Analysis and experimental evaluation of data plane virtualization with Xen. In *ICNS 09: International Conference on Networking and Services*, Valencia, Spain, April 2009.
22. Divakaran DM, Vicat-Blanc Primet P. Channel provisioning in grid overlay networks. In *Workshop on IP QoS and Traffic Control*, December 2007.
23. Cappello F, Caron E, Dayde M *et al.* Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *GRID 05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pp. 99–106, IEEE Computer Society, 2005; 99–106.
24. Bavier A, Feamster N, Huang M, Peterson L, Rexford J. In vini veritas: realistic and controlled network experimentation. In *SIGCOMM 06: Proceedings of the 2006 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ACM, 2006; 3–14.
25. GENI System Overview. The GENI Project Office, September 2008.
26. Feamster N, Gao L, Rexford J. How to lease the Internet in your spare time. *SIGCOMM Computer Communication Review* 2007; **37**(1): 61–64.
27. Bhatia S, Motiwala M, Muhlbauer W, Valancius V, Bavier A, Feamster N, Peterson L, Rexford J. Hosting virtual networks on commodity hardware. *Georgia Tech Computer Science Technical Report GT-CS-07–10*, January 2008.
28. Linux VServers Project. http://linux-vserver.org [3 October 2009].
29. Bavier A, Bowman M, Chun B, Culler D, Karlin S, Muir S, Peterson L, Roscoe T, Spalink T, Wawrzoniak M. Operating system support for planetary-scale network services. In *NSDI04: Proceedings of the 1st Conference on Networked Systems Design and Implementation*, Berkeley, CA. USENIX Association, 2004; 19.
30. He J, Zhang-Shen R, Li Y, Lee C-Y, Rexford J, Chiang M. Davinci: dynamically adaptive virtual networks for a customized internet. In *CoNEXT 08: Proceedings of the 2008 ACM CoNEXT Conference*, ACM, 2008.
31. Egi N, Greenhalgh A, Handley M, Hoerdt M, Huici F, Mathy L. Towards high performance virtual routers on commodity hardware. In *CoNEXT 08: Proceedings of the 2008 ACM CoNEXT conference*, ACM, 2008.

DOI: 10.1002/nem

32. Santos JR, Turner Y, Janakiraman G, Pratt I. Bridging the gap between software and hardware techniques for i/o virtualization. In *ATC08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*, Berkeley, CA. USENIX Association, 2008; 29–42.
33. Wang Y, Keller E, Biskeborn B, van der Merwe J, Rexford J. Virtual routers on the move: live router migration as a network-management primitive. In *SIGCOMM Comput. Commun. Rev.*, 2008; **38**(4): 231–242.
34. Yu M, Yi Y, Rexford J. Rethinking virtual network embedding: substrate support for path splitting and migration. In *SIGCOMM Comput. Commun. Rev.*, 2008; **38**(2): 17–29.

AUTHOR'S BIOGRAPHIES

Fabienne Anhalt graduated from INSA de Lyon in 2008 with a degree in Computer Engineering and received her Master's degree in Computer Science. She is now a Ph.D. student at INRIA within LIP at École Normale Supérieure de Lyon. She has interest on network virtualization, resource sharing and management, and virtual routers.

Guilherme Koslovski is a Ph.D student in Computer Science at École Normale Supérieure de Lyon (ENS, France). He obtained his graduate and Master's (2008) degrees in Computer Science at Federal University of Santa Maria (UFSM, Brazil). He has interest on the following topics: specification, allocation and monitoring of virtual infrastructures.

Pascale Vicat-Blanc Primet, PhD (88) and HDR (02) in Computer Science, is Research Director at the National Institute of Research in Computer Science (INRIA) since 2005. From 1989 to 2004, she was Associate Professor at the École Centrale de Lyon. Since 2002, she leads the INRIA RESO team (22 researchers and engineers). She is a member of the board of Directors of Ecole Normale Supérieure de Lyon and co-director of its Computer Science Laboratory (LIP). Her interests include High-Speed and High-Performance Networks, Internet protocols, Protocols architecture, Quality of Service, network and traffic measurement, Network programmability and virtualization, Grid computing and Grid networking and Communications. Pr. Vicat-Blanc Primet is a member of the scientific 'Networks and Telecoms' expert committee of CNRS (National Research and Science Center—France). She has been a member of the steering committee of Grid'5000—French Computer Science Grid initiative—, and participated to evaluation committees of the French National Research Agency (ANR). She has co-chaired the OGF Data Transport Research Group. She is currently the leader of the ANR HIPCAL project. She (has represented or) represents and leads the activity of INRIA in a variety of International, European and National Grid projects including: IST DataGRID, IST DataTAG, RNTL eToile, ACI GdX, IST EC-GIN, ANR IGTMD CARRIOCAS. She is co-chair of the PFLDnet and ACM Gridnets conference steering committees and participates to numerous international program committees. She has published more than 90 papers in International Journal and Conferences in Networking and Grid computing.

DOI: 10.1002/nem