Exploring the virtual infrastructure service concept in Grid'5000

Pascale Vicat-Blanc Primet
INRIA- ENS Lyon
pascale.primet@ens-lyon.fr

Fabienne Anhalt
INRIA - ENS Lyon
fabienne.anhalt@ens-lyon.fr

Guilherme Koslovski
INRIA - ENS Lyon
guilherme.koslovski@ens-lyon.fr

***Abstract*—The convergence of communication and computation portrays a new vision of the services that the Internet can bring to users. There is an emerging need for isolated and protected virtual resource aggregates composed by the sharing in time and space of a set of physical entities. This paper proposes a flexible and open framework to implement this service offering a dynamic access to virtual private interconnected capacities. We develop the underlying *virtual private execution infrastructure* concept and propose a model to control their underlying networks. We illustrate these ideas by describing the adaptation of our HIPerNET software to the context of the experimental large-scale Grid'5000 platform, thus allowing isolated and customised experiments. The goal of the designed service is to provide computer-science researchers with right-timed and right-sized experimental virtual infrastructures. Preliminary experimental results highlight the potential and challenges of this approach.**

I. Motivations

Today, the usage of the Internet is fundamentally changing. Internet services are constructing data centers of unprecedented scale to offer a large diversity of cloud services for research, data mining, email hosting, maps and other features. This evolution leads to the convergence of communication and computation, and portrays a new vision of the services that the Internet can bring to users. According to this concept, the Internet will not remain "only" a huge shared and unreliable communication facility between edge hosts enabling real time contact and data exchanges. Instead, it will become a worldwide reservoir of interconnected resources that can be shared and reserved. We envision the Internet will increasingly embed and expose its computational and storage resources, as well as its communication and interconnection capacities, in order to be able to meet the requirements of emerging applications.

Large-scale experimental facilities are prefiguring this new way of sharing IT and computing resources and highlight the need for on-demand customizable infrastructures. Indeed, many computer science projects in network or distributed systems require experiments with modified operating systems and communication protocols exposed to realistic and reproducible conditions. Computer scientists need to perform distributed experiments that run on many sites at the same time. Generally the experiments are interactive and large-scale: they run on many nodes, but for a relatively short time (a few hours). This raises the need for time-limited access to customized experimental platforms. As an example, PlanetLab [1] allows researchers to run experiments on a large scale under real-world conditions. Using distributed virtualization, every user can allocate a slice of PlanetLab's network-wide hardware resources for experiments in file sharing and network-embedded storage, content-distribution networks, routing and multicast overlays, network measurement tools, etc. Grid'5000 [2], another experimental facility, gathers large scale clusters and gives access to 5000 CPUs distributed over 9 sites and interconnected by 10 Gbps-dedicated lambdas. Grid'5000 provides a deep reconfiguration mechanism allowing researchers to deploy, install, boot and run their specific software images, possibly including all the layers of the software stack. This reconfiguration capability led to the experiment workflow followed by Grid'5000 users: reserve a partition of Grid'5000, deploy a software image on the reserved nodes, reboot all the machines of the partition using the software image, run the experiment, collect results and release the machines. Grid'5000 allows users to reserve the same set of resources across successive experiments, to run their experiments in dedicated nodes (obtained by reservation), and to install and run their own experimental condition injectors and measurement software. However, predictable connectivity with controlled jitter and dedicated throughput is not currently provided within Grid'5000. These properties can be delivered to the application only through careful direct control of the networking resources as it is proposed in this paper.

We argue that exposing bandwidth as well as processing and storage capacities within the network will help to support the ever-growing spectrum of communication patterns and ways to use the Internet. Extending the approach adopted by researchers in PlanetLab or Grid'5000, we propose the *virtual infrastructure service layer*, to homogeneously decouple the physical infrastructure from the high-level service-plane. In particular, this paper investigates a model and the mechanisms required for flexibly sharing the physical infrastructure of Grid'5000 considering the network backbone as a first-class resource.

Section II defines the *virtual private execution infrastructure* model which portrays a new way of sharing networks and end resources. In Section III, the adaptation of this model to Grid'5000 for virtual experimental infrastructures orchestration is presented. In Section IV preliminary experimental results are given to illustrate the interest as well as technical issues. Related works are reviewed in Section V. Section VI concludes this work.

II. A NEW WAY OF SHARING AND CONTROL

A. *Extending the virtualization concept*

Virtualization enables an efficient separation between services or applications and physical resources. For example, the virtual machine paradigm is becoming a key feature of servers, distributed systems, and grids as it provides a powerful abstraction. It has the potential to simplify the management of resources 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) allows the applications and processes running within a VM to be managed as a whole. Extending these properties to the service level through the concept of "infrastructure as a service", the abstraction of the hardware enables the creation of multiple, isolated, and protected virtual aggregates on the same set of physical resources by sharing them in time and space. In other words, with representation in VMs, it is possible that a physical resource (node) hosts VMs of different virtual infrastructures. The virtual infrastructures are logically isolated by virtualization and can provide customized services to each virtual infrastructure, for example in terms of bandwidth provisioning, channel encryption, addressing, protocol version. The isolation also provides a high security level for each infrastructure. Moreover, virtualizing routers and switching equipments enables the customization of packet routing, packet scheduling and traffic engineering for each virtual network crossing it. The customization of the router's function offers a high flexibility for each infrastructure.

B. *Virtual Private Execution Infrastructures*

In this context, we define the *Virtual Private eXecution Infrastructure (VPXI)* concept as the aggregation of virtual computing resources interconnected by a virtual private overlay network. Ideally, any user of a VPXI has the illusion that he is using his own dedicated system, while in reality he 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 TCP/IP applications he was using on the Internet or its Intranet.

A VPXI can be formally represented as a graph in which a vertex is in charge of active data processing functions and an edge in charge of moving data between vertices. Figure 1 illustrates this concept, representing a virtual infrastructure composed by the aggregation of virtual machines interconnected via virtual channels. It shows two virtual routers (vertices r_vA and r_vB) which are used to interconnect and perform the network control functions among the other virtual resources (vertices r_v 1 to 8). The virtual routers can independently forward the traffic of the different virtual infrastructures which share the same physical network. Each edge represents a virtual link used to interconnect a pair of virtual resources, which contains different configurations, as l_v1 and l_v2.

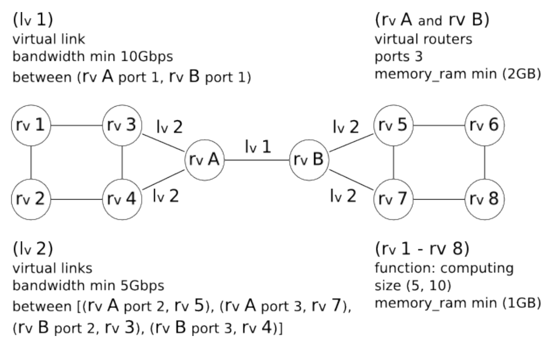

Figure 1. Example of a VPXI composition using graph notation.

A VPXI specification comprises the recursive description of: a) the individual end resources or resource aggregates (clusters) involved, b) the performance attributes for each resource element (capacity), c) the security attributes for each resource element (access control, confidentiality level), c) the commercial attributes for each resource element (maximum cost), d) the temporal attributes for each resource element (time window for provisioning), e) elementary functions, which can be attributed to a single resource or a cluster, for example: request of *computing* nodes, *storage* nodes, *visualization* nodes, or *routing* nodes, f) the specific services provided by the resource (data mining application, data compression software), g) the topology of the virtual network, including the performance characteristics (typically bandwidth and latency), the security, commercial, and temporal attributes of the virtual channels. A VPXI has a limited lifetime which can span from a few hours to several months. To support the specifications of these VPXI (virtual environments), the VXDL language has been studied and developed [3].

C. *VPXRouters*

Within the VPXI design, we propose virtual routers called VPXRouters which are fully personalizable components of the VPXIs. These virtual routers are hosted on open high-performance physical servers used as software routers (that we call HPSRouters, for High Performance Software Routers), each one running in an isolated virtual machine instance. In this approach, all the traditional network planes (data, control and management) are virtualized. Therefore users can use any protocol and control mechanism on their allocated virtual routers. They can deploy customized routing protocols, configure the packet-queueing disciplines, packet filtering and monitoring mechanisms they want. Also, VPXRouters represent strategic points of the network for rate control as they concentrate aggregated VPXI traffic. By limiting the rate and policing the traffic at the VPXRouters, the traffic of VPXIs can be controlled and the user is provided with fully- isolated execution infrastructures. The benefit of having controlled environments is twofold: it gives the users strong guarantees, while allowing the network provider to better exploit the network by sharing it efficiently, but differently, between users.

D. Embedding Virtual Infrastructures

Using VXDL language users can specify the desirable configuration and network composition of VPXIs. This request must be interpreted and reserved on available distributed resources. This virtual infrastructure composition corresponds to a graph embedding problem, where a graph which describes the virtual infrastructure must be allocated on a graph which describes the physical infrastructure. As it is not the purpose of this paper, the general embedding problem is not studied here but just presented.

Virtual and physical graphs are of the form $G(V, E)$ where vertices V are a set of resources interconnected by a set of links (edges represented by E). We consider that each resource or link can have a capacity represented by c_v and c_p for virtual and physical components respectively. Capacities can be interpreted as configurations like latency and bandwidth for links, or memory size, CPU speed, and physical location for resources. To illustrate the embedding of a VPXI, we select the *VPXI 2* from figure 4 and create a possible specification (as presented in Figure 2.a) which represents three virtual resources (r_v1, r_v2 and r_v3) and three virtual routers (r_vA, r_vB and r_vC), interconnected by some virtual links ($l_v1, l_v2, l_v3, l_v4, l_v5$ and l_v6). In this scenario, every resource and link has a different capacity. Analyzing Figure 2.b, we can

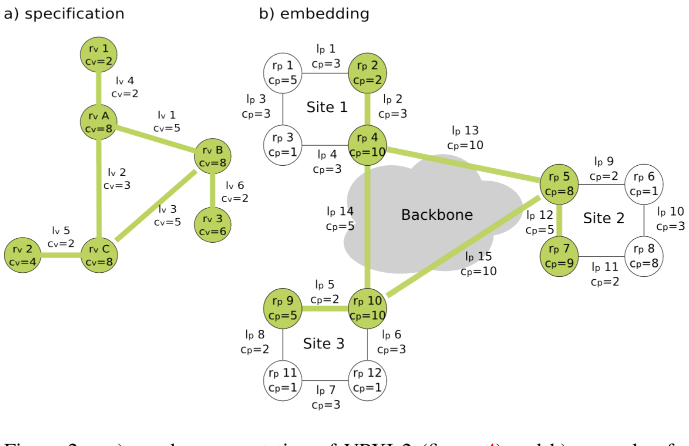

Figure 2. a) graph representation of VPXI 2 (figure 4) and b) example of an embedding solution.

observe an embedding alternative for *VPXI 2*. In this case, each site received one virtual router and one virtual resource. The network topology specified in 2.a was also allocated. This scenario illustrates a VPXI specification which has different network capacities ($c_v = 2, c_v = 3$ and $c_v = 5$). These capacities can correspond to different bandwidth specifications that must be allocated and controlled.

The result of the preceding embedding example is annotated in a map form as presented in table I. The first line represents

Component	Embedding
resources	$< r_v1, r_p2, 2, \Delta t_0 >$
	$< r_v2, r_p9, 4, \Delta t_0 >$
	$< r_v3, r_p7, 6, \Delta t_0 >$
	$< r_vA, r_p4, 8, \Delta t_0 >$
	$< r_vB, r_p5, 8, \Delta t_0 >$
	$< r_vC, r_p10, 8, \Delta t_0 >$
links	$< l_v1, l_p13, 5, \Delta t_0 >$
	$< l_v2, l_p14, 3, \Delta t_0 >$
	$< l_v3, l_p15, 5, \Delta t_0 >$
	$< l_v4, l_p2, 2, \Delta t_0 >$
	$< l_v5, l_p5, 2, \Delta t_0 >$
	$< l_v6, l_p12, 2, \Delta t_0 >$

Table I
EMBEDDING SOLUTION FOR THE EXAMPLE IN FIGURE 2.

resources notation and the second one represents links notation. In this example, only one period $\Delta t0$ was used which means that all the resources and links must be reserved and used at the same time with a defined duration.

III. ADAPTATION TO GRID'5000 EXPERIMENTS

To allow users to specify, reserve, deploy and manage *virtual private execution infrastructures* over large-scale distributed systems, we are developing the HIPerNET[1] [4] software. We describe here how the HIPerNET software is currently being adapted to Grid'5000 [2], the national experimental shared facility, to enable reproducible experiments on customizable topologies. Figure 3 represents the Grid'5000

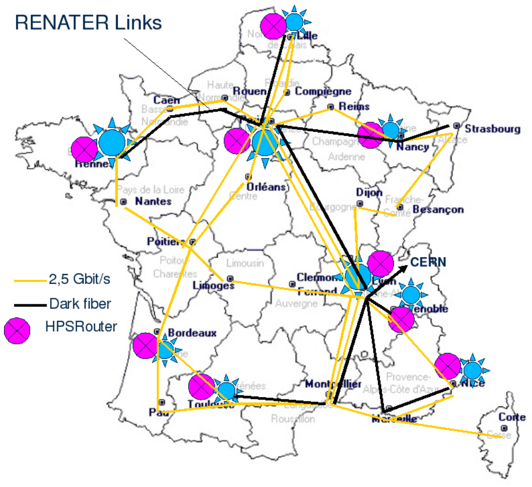

Figure 3. Grid'5000 infrastructure with HPSRouters.

testbed with its nine sites interconnected with 10 Gb/s dedicated lambdas. An HPSRouter is inserted on each site. These machines host VPXI's VPXRouters to support diverse routing strategies and innovative transport protocols. Another goal is to control the bandwidth-sharing of the physical inter-site links for isolated, controlled, and reproducible experiments. VPXIs

[1]http://www.ens-lyon.fr/LIP/RESO/Projects/HIPCAL/ProjetsHIPCAL.html

can be reserved and deployed over several geographically distributed sites. Figure 4 represents an example of three VPXIs, extended over three distinct sites. Each site is provided with a HPSRouter hosting one VPXRouter per VPXI. Those virtual routers are gateways, forwarding all the traffic of a VPXI between the site's LAN and the interconnection network. With

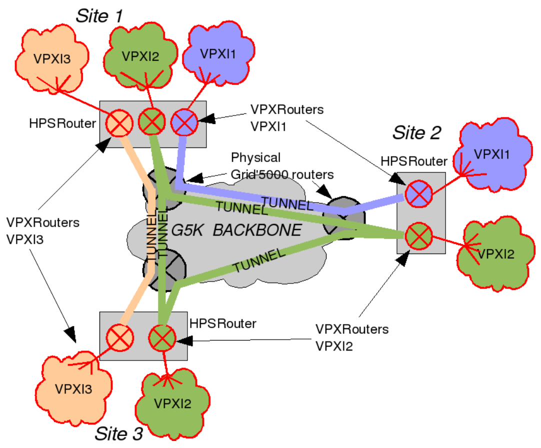

Figure 4. Allocation of 3 VPXIs in the Grid'5000 infrastructure.

this model, the VPXRouters are able to control the traffic of the VPXIs and their bandwidth sharing of the physical links over the backbone network. The VPXRouters are interconnected over the backbone across several hops via IP-tunnels giving the VPXI user the illusion that the different parts of his VPXI are directly interconnected by a single router, even though they are in reality located on distant physical locations.
To face scalability issues, we assume that at a given time, only a limited number of experiments will request for a confined VPXI and a dedicated channel (for example, 10%=1 Gb/s). The others, which are considered not "communication sensitive", will run without any network control in the classical and fully-transparent best effort mode. The aggregated bandwidth allocated to the VPXI is limited to a configurable percentage of the access link's capacity. The shaped VPXI-traffic leaving the physical routers hosting the VPXRouters is fully isolated from the remaining best-effort traffic of Grid'5000. To guarantee this, the switch where the VPXRouter-traffic and the Grid'5000 best-effort traffic come together distinguishes the two traffics and gives a strict priority to the VPXRouter-traffic. The background traffic, i.e. all the traffic that does not require specific treatment, is forwarded through the classical best effort path.

The following sections detail the different building blocks for the implementation of this model.

A. Virtual Router

During the deployment of a VPXI, a VPXRouter is created and started on each dedicated physical router located on a site hosting some of the VPXI's virtual resources.

1) VPXRouter Architecture: A VPXRouter consists of a high performance physical machine (HPSRouter) running Xen and owning two 10 Gb/s network interface cards. The VPXRouters consist in software routers implemented inside virtual machines. We chose the Xen 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. Figure 5 shows an example of an HPSRouter hosting two VPXRouters. These VPXRouters have to share the physical resources of the machine and additional processing is necessary. However, the data plane virtualization with Xen causes the forwarded packets to go through a longer path than in native-Linux software routers.

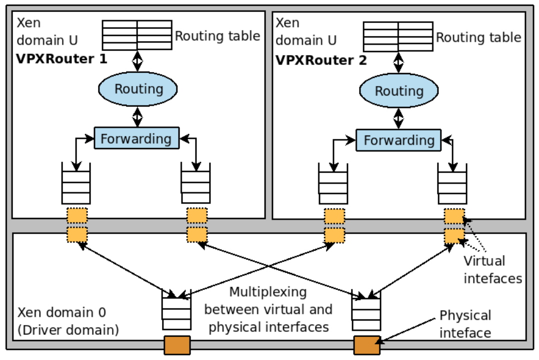

Figure 5. Model of an HPSRouter hosting two VPXRouters.

2) Virtual routing: VPXRouters control the rate and adapt the routing in order to satisfy the quality of service requirements of each VPXI, as illustrated in the example below(Figure 6). Latency sensitive traffic and bandwidth-aware

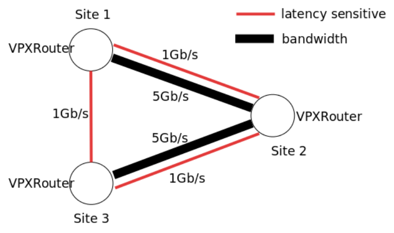

Figure 6. Example of bandwidth allocation for latency-sensitive and high-bandwidth flows.

traffic are routed on different paths. 1 Gb/s links are allocated between each of the three VPXRouters to transmit latency sensitive traffic. High-throughput traffic is redirected over Site 2, requiring to allocate additional 5 Gb/s on only 2 links instead of 3. As presented in [5], this is an example of efficient channel provisioning combining routing and traffic engineering.

B. Rate allocation

To give predictable and reproducible results, all the specified virtual resources requested for an experiment need to be allocated and reserved. Users can specify the necessary resources for their VPXIs like CPU, memory, but also the rate and latency of the interconnection links. In the Grid'5000 case, we manage virtual links over the RENATER backbone and reserve access link capacities.

1) User services: In the Grid'5000 HIPerNET model, three bandwidth reservation services are offered :

- **Guaranteed minimum:** The minimum rate that should be available in the VPXI at any moment.
- **Allowed maximum:** The maximum capacity the VPXI should allow at any moment.
- **Static reservation:** In this case, the guaranteed minimum is equal to the allowed maximum.

These services can be used to different ends. For example, a user who wants to execute a distributed application communicating with MPI will specify a guaranteed minimum rate. Limiting the traffic of a VPXI to a maximum rate gives the user the impression that he is using a physical network with a maximum link speed. This link is shared in a best-effort way with other traffic. Specifying a static rate gives the user the impression to work in a dedicated physical network whose links have the specified capacity. He will be able to obtain the specified bandwidth at any moment but could never exceed it. Among other things, this kind of service could help making reproducible experiments where link availability should not vary.

At each incoming VPXI-request (new resource specification), HIPerNET determines if the VPXI can be mapped to the physical underlay in a way to guarantee the required minimum bandwidth on all the links, respecting also the topology and other resource specifications. If this is not the case, an alternative VPXI is proposed. When reservation starts, rate control is activated or reconfigured to 1) guarantee the desired minimum rate and/or 2) limit the rate to the desired maximum for each VPXI.

To guarantee a minimum rate on a physical link, all the concurrent virtual links using it are limited.

Let C be the capacity of the link, N the number of virtual links sharing it, $m_{req}(i)$ and $M_{req}(i)$ be respectively the minimum and the maximum requested bandwidth of VPXI i. Let $M_{alloc}(i)$ be the the allocated maximum bandwidth for VPXI i on the considered link; the objective function is to maximize $\sum_{i=1}^{N} M_{alloc}(i)$
subject to

- $m_{req}(i) \leq C - \sum_{j\in[1,N],j\neq i} M_{alloc}(j), \forall i \in [1,N]$
- $m_{req}(i) \leq M_{alloc}(i) \leq M_{req}(i), \forall i \in [1,N]$
- $\sum_{i=1}^{N} m_{req}(i) \leq C$

Given the user specifications for each VPXI i sharing the link, $m_{req}(i)$ and $M_{req}(i)$ (by default, $m_{req}(i) = 0$ and $M_{req}(i) = C$), the optimum values for the maximum bandwidth ($M_{alloc}(i)$) per VPXI are calculated. The following example will illustrate such an allocation scheme: Having a 10 Gb/s physical link on each site, 2 Gb/s are reserved for the VPXIs, while the remaining bandwidth (8 Gb/s) are used for best-effort traffic. Three VPXIs share the physical link and the HPSRouter of each site has three VPXRouters. Let's assume each VPXI's user specifies the same bandwidth for all the VPXI's virtual links. For the virtual links of VPXI 1 and 2, the users request respectively a minium (m_{req}) of $100Mb/s$ and $800Mb/s$ and a maximum (M_{req}) of $500Mb/s$ and $1500Mb/s$. The user of VPXI 3 makes a static reservation of $m_{req} = M_{req} = 300Mb/s$ for all the virtual links. At timestamp t_1, all of those three VPXIs are running. At timestamp t_2, VPXI 3 finishes its timeline and only VPXI 1 and 2 remain, so the rate allocations are recalculated. The resulting allocation is illustrated in Table II. During the first phase, from t_1 to t_2, all the virtual links can get the desired minimum rate, but it is not possible to allocate the maximum desired rate for each one. So the links of VPXI 2 can attempt only a rate of 1200 Mb/s instead of 1500 Mb/s. At t_2, VPXI 3 finishes, and the links of VPXI 1 and 2 can share the remaining bandwidth. From this moment on, both VPXI 1 and 2 can use their maximum desired bandwidth on the specified virtual links.

		VPXI 1	VPXI 2	VPXI 3
t_1	m_{req}	100	800	300
VPXI 1-3	M_{req}	500	1500	300
allocated	M_{alloc}	500	1200	300
t_2	m_{req}	100	800	
VPXI 1-2	M_{req}	500	1500	
allocated	M_{alloc}	500	1500	

Table II
RATE ALLOCATED PER VPXI (MB/S).

2) Rate control techniques: A variety of technologies, especially those provided by the Linux `traffic control`[2] tool, can be applied to control the rate on the HPSRouters. Different locations can be identified inside the HPSRouter where rate control can take place. Considering the path of a packet through the HPSRouter as represented on Figure 7, there are four possible places on this path to implement traffic control:

1) At the incoming on the physical interface of the HPSRouter(dom0);
2) At the outgoing on the virtual interface of dom0;
3) At the outgoing on the virtual interface of the VPXRouter(domU);
4) At the outgoing on the physical interface of the HPSRouter(dom0).

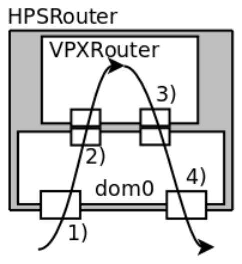

Figure 7. Potential locations of rate-control mechanisms.

Limiting at the incoming physical interface (1) would consist in dropping packets when the allocated bandwidth is exceeded. This solution is unsatisfactory because shaping the traffic is

[2]http://lartc.org

preferable, in order to lose as few packets as possible. Limiting at the outgoing virtual interfaces of dom0 seems to be a good solution, as the traffic is shaped as soon as possible, before even entering the virtual routers. Limiting at the outgoing virtual interface of the virtual routers would be a solution too, but shaping as soon as possible, i.e. before entering the virtual router, would be preferable. The last shaping opportunity occurs at the outgoing physical interface of dom0 (4), before the packets leave the interface. The advantage, compared with solutions 1, 2, and 3, is that the shaping function knows the traffic of all the VPXIs, since it is concentrated on this interface. This could help in adapting the treatment of one VPXI's traffic according to that of the other VPXIs. We thus focus on limiting the traffic at the outgoing interfaces of dom0, either the virtual ones or the physical ones. To shape the traffic on the physical interface, a classful queueing discipline is required in order to treat the traffic of each VPXI in a different class. Limiting at the virtual interfaces would mean that each virtual router has its own queueing discipline and so no classful qdiscs are needed.

IV. Experimental results

This section presents experimental results about virtual router performance and bandwidth control. All the experiments are executed within the Grid'5000 [2] platform, using Xen 3.2 and IBM opteron servers with either one or two 1 Gb/s physical interfaces. The machines have two CPUs with one core each. Reference results are obtained using Linux with a 2.6.18 kernel.

A. Xen Virtual Router performance

As described before, virtualizing the data plane with Xen introduces a longer path for the packets to go through, and the physical resources have to be shared between several virtual routers. In these experiments, the throughput, latency, and scalability of VPXRouters are evaluated as well as the resulting CPU overhead on Xen virtual routers. TCP and UDP flows are sent over 1, 2, 4, or 8 virtual routers, sharing a single physical machine with two physical network interfaces. All the senders, routers and receivers are interconnected by one switch; native-Linux throughput corresponds to the theoretical values (941 Mb/s with TCP and 952 Mb/s with UDP) and latency to around 0,017 ms. The measured TCP and UDP rates with big packets (1500 bytes) are represented on Figure 8. The measured UDP receive rate reaches the theoretical value (952 Mb/s), whether using a single virtual router or eight virtual routers at a time. The increasing rate with an increasing number of virtual routers is due to little variations between the start times of the flows. Packet loss is only related to the sharing of the network interfaces, which is fair and using the maximum capacity. With TCP, the results show that the throughput is affected by the virtualization. It reaches only about 85% of its theoretical throughput of 941 Mb/s for one virtual router, and even less as the number of concurrent virtual routers increases. Table III shows that this latency increases

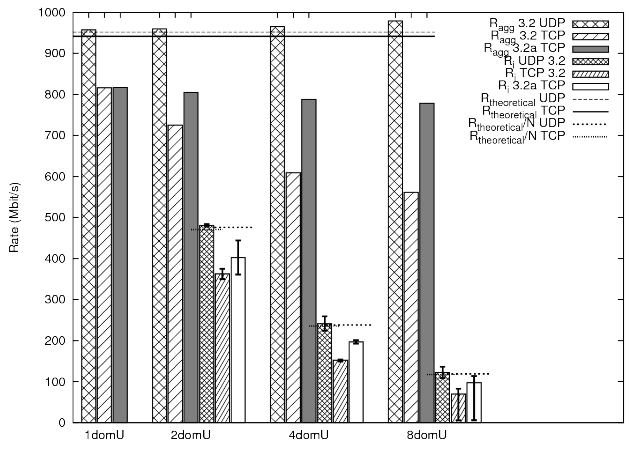

Figure 8. Receiver-side throughput over 1, 2, 4, or 8 VPXRouters.

Latency(ms)	linux	1 VR	2 VR	4 VR	8 VR
idle	0.084	0.147	0.150	0.147	0.154
stressed			0.888	1.376	3.8515

Table III
LATENCY OVER ONE VPXROUTER (VR) AMONG 1, 2, 4, AND 8, BEING IDLE OR STRESSED WITH TCP FORWARDING.

significantly, as we increase the number of virtual routers forwarding TCP flows. The TCP throughput could be increased by giving more CPU scheduler weight to dom0 (result 3.2a), which would also decrease the latency. The good performance in UDP and the small overhead with TCP (the resulting throughput reaches about 84% of the native Linux throughput) show that the virtual-router approach is a promising idea for the VPXRouter model, becoming scalable and efficient. Even better performance can be expected considering the evolution of virtualization techniques and hardware.

B. Rate control

The goal of this experiment is to evaluate the behaviour of classical rate-control mechanisms in the context of virtualization, on the routers hosting the VPXRouters. Three

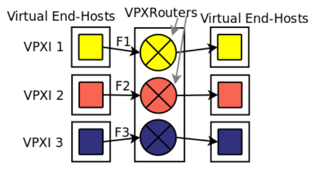

Figure 9. Experimental setup with 3 VPXRouters controlling rates for 3 VPXIs.

VPXRouters, hosted by a single HPSRouter, control the rate of three different VPXIs (Figure 9). The rates allocated (R_{alloc}) on the VPXRouters of VPXI 1, 2, and 3 are respectively 100 Mb/s, 150 Mb/s and 200 Mb/s. Three flows (F1, F2 and F3), are sent over the three VPXRouters with different input rates(R_{input}) to vary the congestion factor $CF = R_{input}/R_{alloc}$. To generate, for example, a normal load (CF=0.9), the input rates of the flows F1, F2 and F3 are

respectively 90 Mb/s, 135 Mb/s and 180 Mb/s.
In these experiments, three scenarios are considered to com-

				Linux		Virtualization	
			R_{alloc}	Rate	Loss	Rate	loss
CF=1	PSP	F1	100	96.5	3.8%	96.4	3.9%
		F2	150	145	3.8%	145	3.9%
		F3	200	193	4.7%	192	4.8%
	TBF on NIC	F1	100	98.7	1.6%	98.0	2.4%
		F2	150	149	1.3%	147	2.4%
		F3	200	197	2.5%	196	3.2%
	TBF on VIF	F1	100			97.9	2.5%
		F2	150			147	2.4%
		F3	200			196	3.2%
CF=0.9	PSP	F1	100	90.4	0.052%	90.3	0.15%
		F2	150	135	0.058%	135	0.16%
		F3	200	181	0.057%	181	0.17%
	TBF on NIC	F1	100	90.4	0.013%	90.4	0.08%
		F2	150	135	0.043%	135	0.1%
		F3	200	181	0.037%	181	0.12%
	TBF on VIF	F1	100			90.4	0.018%
		F2	150			135	0.02%
		F3	200			181	0.03%

Table IV
UDP RECEIVE RATE (MB/S) AND LOSS (%) WITH DIFFERENT CONGESTION FACTORS (CF).

pare traffic shaping with PSPacer to traffic shaping with the tocken bucket filter (TBF qdisc): 1) PSPacer is implemented on the phsyical interfaces of the router; 2) a `prio` qdisc is used on the physical interface (NIC) to hold a TBF in each of its three classes; 3) a TBF is implemented on each virtual interface(Figure 7.2).

Table IV shows the UDP rate R_{output} obtained by the flows F1, F2 and F3 with a congestion factor CF of 1 (limit load) and 0.9 (normal load). With $CF = 1$, small losses can be observed in all the configurations, whereas the loss is slightly higher with virtualization. Also PSPacer shows a little more loss than TBF. With $CF = 0.9$ and no virtualization, the loss is about 0 for all the configurations. It is slightly higher with virtualization but still very close to 0 (<0.2%). The TBF implemented on the virtual interfaces (Figure 7.2)) shows the smallest loss (<0.04%). This good result is probably related to the fact that the TBF is implemented on the entrance of the virtual routers, limiting the rate as soon as possible.
Figures 10, 11 and 12 represent the TCP throughput over the three virtual routers implementing the described rate control. Table V shows the TCP throughput on a classical Linux router without virtualization, forwarding three flows and applying the same rate-control mechanisms. These are average values; the rate variation is insignificant over the 60s measurement interval. Compared to linux, virtualization has an impact

	F1 (Mb/s)	F2 (Mb/s)	F3 (Mb/s)
PSP	86.6	130	174
TBF	86.6	130	174

Table V
TCP RATE ON LINUX WITH A CONGESTION FACTOR OF 0.9.

on the rate control mechanisms. Especially for PSPacer, the

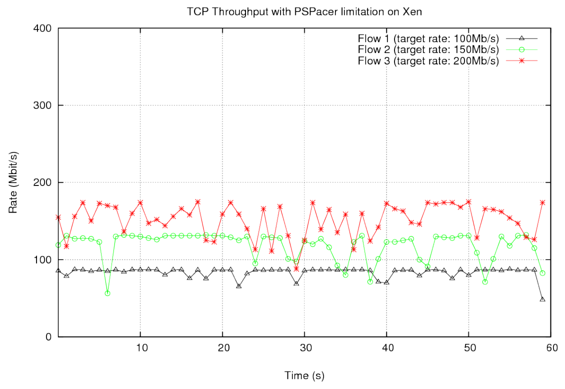

Figure 10. TCP Rate over three virtual routers with PSPacer.

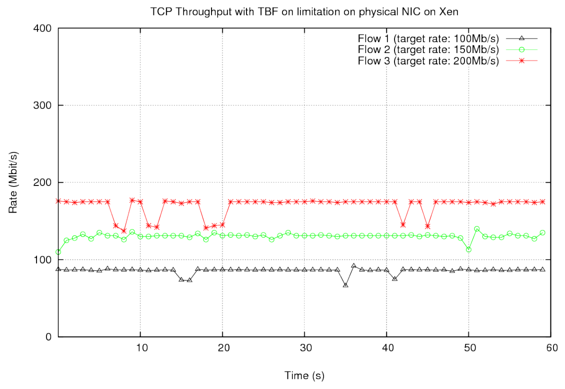

Figure 11. TCP Rate over three virtual routers with TBFs in a prio qdisc.

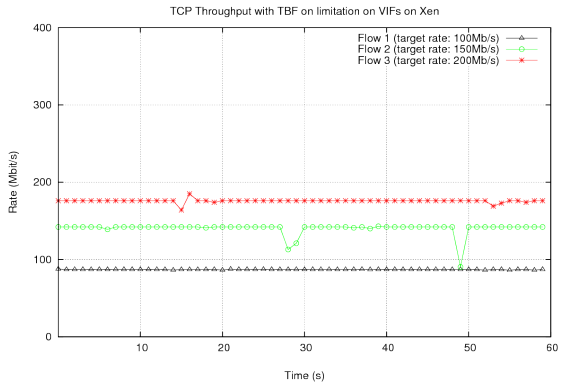

Figure 12. TCP Rate over three virtual routers with TBF on VIFs.

throughput is impacted by the virtualization for big rates. With TBF, the throughput varies less and is bounded to the input rate, but the results are still more predictable when TBF is implemented on the virtual interfaces, controlling the flows before entering the virtual routers. The overall performance is decreased slightly with TCP on the virtual routers, as shown by the previous experiment (IV-A), but is still promising, achieving a rate close to the desired values.

V. Related work

The approach of controlled virtual network infrastructures, running in parallel over a shared physical network is an emerging idea offering a variety of new features for the network. VINI [6] is a virtual network infrastructure where researchers can run experiments in virtual network slices running XORP routing software inside UML instances they can configure and chose between the available protocols. HIPerNET pushes this facility a step further, adding data-plane virtualization and allowing the user to chose the operating system and install any routing software. This provides full isolation between virtual nodes.

The CABO [7] design describes how to decouple services from infrastructure using virtualization, to give Internet service providers end-to-end control while using the physical equipments of different physical infrastructure providers. The HIPerNET implementation focuses more on the combination of network with end-host virtualization to provide the users with controlled virtual computing infrastructures.

In the GENI design [8], users are provided with slices composed by either virtual resources or partitions of physical resources. Its goal is rather to provide the user with multiple shareable types of resources with high but limited reconfigurability being programmable and software can be uploaded. The main difference between these projects and HIPerNET is that HIPerNET provides the user with *full reconfigurability*, just as in Grid'5000, where users can deploy any operating system of their choice. This opposes Grid'5000 to other platforms like PlanetLab [1], where pre-installed slices can be reserved to execute user software.

DaVinci [9] uses virtual networks to isolate traffic classes and run different traffic management protocols like in HIPerNET's VXRouters. Virtual networks are dynamically and periodically adapted to optimize link utilization, while HIPerNET updates the link allocations at each new or ending request, focusing on the policing and control of the substrat sharing.

Virtual routers have improved their performance over the last years [10]. However, Xen's data-plane virtualization impacts the performance [11]. But as HIPerNET focuses on full reconfigurability, including OS choice, the data-plane virtualization is of interest. Also, Xen's performance have been growing with successive versions [12]. Current hardware solutions do not allow the same reconfiguration level than the VPXRouters. For example Juniper's TX Matrix Plus routers[13] allow to run up to 16 routing instances, virtualizing the control-plane, what offers above all more routing capacity for less power like in server consolidation.

VI. Conclusion

Considering the convergence of the communication, computation and storage aspects of the Internet, this paper advocates for the design, development and deployment of new resource-management approaches to discover, reserve, co-allocate and reconfigure resources, schedule and control their usages. This paper developed the virtual private execution infrastructure concept to offer advanced IT service providers a dynamic access to extensible virtual private capacities, through on-demand and in-advance bandwidth- and resource-reservation services. This paper studied in particular an adaptation of the virtual infrastructure concept and the HIPerNET software to the Grid5000 facility. The goal is to provide users with fully confined environment and enable reproducible experiments which is not the case for other testbeds. We have prototyped a software virtual router model which virtualizes both data and control planes. Our results show that performances are promising. Traffic is managed within each VPXI with classical Linux traffic control mechanisms so that the users obtain fully isolated channels where they can route freely their traffic.

Acknowledgement

This work has been funded by the French ministry of Education and Research and the ANR, INRIA, and CNRS, via ACI GRID's Grid'5000 project, ANR HIPCAL grant, INRIA Aladdin ADT and the CARRIOCAS project (Pôle SYSTEM@TIC IdF).

References

[1] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating system support for planetary-scale network services," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, (Berkeley, CA, USA), pp. 19–19, USENIX Association, 2004.

[2] F. Cappello, P. Primet 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.

[3] G. P. Koslovski, P. Vicat-Blanc Primet, and A. S. Charão, "VXDL: Virtual Resources and Interconnection Networks Description Language," in *GridNets 2008*, Oct. 2008.

[4] J. Laganier and P. Vicat-Blanc Primet, "Hipernet: a decentralized security infrastructure for large scale grid environments," in *6th IEEE/ACM International Conference on Grid Computing (GRID 2005), November 13-14, 2005, Seattle, Washington, USA, Proceedings*, pp. 140–147, IEEE, 2005.

[5] D. M. Divakaran and P. Vicat-Blanc Primet, "Channel Provisioning in Grid Overlay Networks (short paper)," in *Workshop on IP QoS and Traffic Control*, Dec 2007.

[6] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "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*, pp. 3–14, ACM, 2006.

[7] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, 2007.

[8] "GENI System Overview." The GENI Project Office, September 2008.

[9] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "Davinci: Dynamically adaptive virtual networks for a customized internet," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT conference*, ACM, 2008.

[10] A. Menon, A. L. Cox, and W. Zwaenepoel, "Optimizing network virtualization in xen," in *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pp. 2–2, USENIX Association, 2006.

[11] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Towards high performance virtual routers on commodity hardware," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT conference*, ACM, 2008.

[12] F. Anhalt and P. Vicat-Blanc Primet, "Analysis and experimental evaluation of data plane virtualization with Xen," in *ICNS 09 : International Conference on Networking and Services*, (Valencia, Spain), Apr. 2009.

[13] "http://www.juniper.net/products_and_services/t_series_core_platforms/index.html."