VIPER: Fine Control of Resource Sharing in Virtual Networks

Natalia Castro Fernandes and Otto Carlos Muniz Bandeira Duarte
GTA/COPPE - Universidade Federal do Rio de Janeiro - Rio de Janeiro, Brasil

***Abstract*—We propose, implement, and evaluate VIPER, a system to isolate, provide QoS, and manage virtual networks. Contrary to previous approaches, VIPER guarantees a fine sharing of physical resources among virtual networks according to the different parameters that describe the service level agreements. The main components of the proposed system are the *resource sharing manager* and the *virtual network admission controller*. The resource sharing manager achieves isolation by dynamically adapting itself to the resource demands of each virtual network. Based on the data monitored by the resource sharing manager, the virtual network admission controller builds network profiles that serve as the basis for arbitrating the access of new virtual networks to the physical substrate. VIPER also supports two levels of QoS control, one for the virtual network operator and the other for the infrastructure provider, reducing delays by up to 18 times in the analyzed scenarios. We developed a prototype whose evaluation reveals that VIPER, when compared to the other solutions in the literature: (i) enforces contracted agreements, (ii) provides an efficient admission control of new virtual networks, and (iii) reduces physical resource utilization by up to 25%.**

I. Introduction

Virtualization technologies enable the splitting of a physical network into slices, called virtual networks, each with its own protocol stack and addressing scheme [1]. Virtualization, therefore, introduces more flexibility into the network core, supporting innovation. The implementation of virtual networks requires, however, three main features that are not provided by proposals that focus on the control of resource sharing among virtual machines in data centers [2], [3]. These features are: isolation, to ensure that virtual networks hosted in the same physical hardware do not interfere with each other; high packet forwarding performance, guaranteeing that virtual routers are profitable, presenting an efficiency similar to physical routers; and quality-of-service (QoS), to surpass virtualization layer constraints and foster the deployment of new applications with bandwidth and delay requirements [4].

Providing virtual network isolation depends on a *fair resource sharing* of the physical routers. Resource demands of virtual routers for CPU, memory, and bandwidth, however, vary with time, hindering the provision of the service level agreements (SLAs). This imposes two requirements. First, it is necessary that the virtual network slices *automatically adapt* to the demands according to the SLAs. Second, we must *control the number* of virtual routers hosted in the same physical machine in order to limit the probability that simultaneous peaks of resource demand violate the contracts.

In order to provide these challenging functionalities, we propose VIPER, the Virtual network Isolation, Policy Enforcement, and Resource sharing system for control and resource management of virtual networks. VIPER provides physical resource sharing mechanisms based on an efficient adaptive slicing mechanism. These mechanisms guarantee strong isolation among virtual networks while totally respecting the SLAs. VIPER also supports options for improving the packet forwarding performance, by using a shared data plane, and for establishing QoS primitives to differentiate the traffic inside a virtual network and among virtual networks. VIPER architecture is composed of two main building blocks:

- **Resource sharing manager -** It monitors the traffic of each virtual network, adaptively adjusts the resource allocation parameters, and punishes the virtual networks that violate the SLAs. Hence, the resource manager ensures a correct slicing based on the resource demands.
- **Virtual network admission controller -** It verifies whether there are enough physical resources to host a new virtual network without restricting the performance of well-behaved virtual networks hosted by the physical router. The admission control algorithm provides an accurate prediction of long-term demands.

We developed a prototype of VIPER on the Xen virtualization platform [5], [6]. The results show that VIPER ensures high conformity of the forwarded traffic to the characteristics specified in SLAs. Besides, our proposal guarantees a more efficient link usage when compared to different configurations of some of the best existing solutions, namely Traffic Control tool [7] and Xen Network Monitor [8]. The tests to assess isolation and the provision of SLA guarantees show that VIPER reduces by up to 18 times the delays of traffic with priority requirements when compared to systems without QoS support in the infrastructure provider. In addition, the system outperforms by five times the other tools in provision of SLA guarantees. We also performed simulations with different resource demand patterns to evaluate the proposed virtual network admission controller. Due to the introduced techniques for estimating demand increases, VIPER overcomes the resource utilization efficiency of other admission control techniques, such as Sandpiper [2].

As a summary, this paper presents the following key contributions: (i) the proposal and development of an efficient resource sharing manager that guarantees the isolation and the SLA provision to virtual networks; (ii) the introduction of new techniques for estimating resource demands that guarantee an accurate virtual network admission control; and (iii) the provision of QoS primitives in virtualized environments.

The remainder of the paper is organized as follows. In Section II, we describe the virtual network model. In Section III, we present the overall architecture of the proposed system and detail it in Sections IV, V, and VI. We then present in Sections VII and VIII, respectively, the simulations results and experimental evaluation. Related works are presented in Section IX. Finally, in Section X, we conclude the paper.

II. Background and challenges

Virtualization software allows network innovation as each virtual environment can be configured as a different router sharing both the physical machine and the physical links. Although VIPER works in any virtualization platform[1], we assume the model of machine virtualization that is used in Xen, as depicted in Fig. 1. In our system model, each virtual machine is created by the infrastructure provider. The infrastructure provider is the entity that controls the physical infrastructure, selling slices to the virtual network operators [12]. Hence, a virtual network is a slice of the physical network, defined as a set of virtual routers and the respective virtual links. This model can be expanded to consider also other network components, such as middle boxes and servers, since they could be hosted inside any virtual machine.

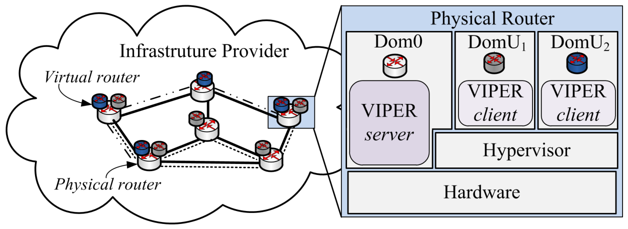


Fig. 1. General view of VIPER architecture applied to Xen, assuming two virtual networks, where each DomU is a virtual router.

Xen virtualizes the whole machine, which implies that a virtual environment supports any operating system and any protocol stack. Xen, however, has problems to ensure I/O isolation between virtual networks, as well as to provide packet forwarding with high-performance [4], [5], [8]. These problems occur because the sharing of physical resources used for I/O operations depends on a *privileged virtual machine* called Domain 0 (Dom0). The Dom0 is a management interface between the user and the hypervisor. In addition, Dom0 schedules virtual machines access to the physical hardware, because it is the only virtual machine with direct access to the hardware. Thus, Dom0 is the driver domain, which maps the virtual device drivers placed in the *unprivileged domains* (DomU) onto the physical device drivers. Since the hypervisor does not control the Dom0 resource usage by each DomU, a DomU can affect the I/O operations of other DomUs by exhausting Dom0's resources. This is a vulnerability of Xen, because the isolation between the networks is impaired.

[1]Each platform presents a different virtualization model, which defines, for instance, if the virtual environment is a complete machine, such as in Xen and in VMware [9], or if all virtual environments share the same operating system, such as in OpenVZ [10] and VServer [11]. Porting VIPER to other substrates would require then a few adjustments for monitoring resource usage and applying the punishments. The foundation, however, remains the same.

The packet forwarding performance in Xen is another challenge. Since a DomU cannot directly access the hardware, the throughput of virtual routers is much lower than the throughput achieved with non-virtualized systems [4]. A solution to this problem is the paradigm of *plane separation* [5]. In this paradigm, the Dom0 is in charge of forwarding the packets of all virtual networks (as Dom0 has direct access to the hardware), obtaining a throughput similar to non-virtualized systems, while each DomU controls its own virtual network. For this purpose, a replica of the data plane generated inside each virtual machine must be maintained in Dom0. Therefore, the plane separation guarantees improved performance in the packet forwarding. On the other hand, the virtual networks lose flexibility, because all networks share a single data plane with the same primitives. As we will see in the following, VIPER overcomes these challenges, balancing efficiency and flexibility to provide a profitable router virtualization model.

III. VIPER: Principles and general architecture

VIPER is a system to control and manage virtual networks. Its main objectives are the isolation of virtual networks by controlling the use of shared resources, such as CPU, memory and bandwidth, and the provision of QoS guarantees. VIPER guarantees robust isolation among virtual routers and differentiates physical resource usage by each virtual router in accordance with the SLAs. Besides monitoring SLA violations, VIPER controls the access of new virtual networks to the physical machines and provides QoS primitives for configuring the virtual environments.

The combination of VIPER and Xen is detailed in Figure 2. The main components are the *resource sharing manager*, which monitors and punishes virtual networks to ensure the isolation, and the *admission controller*. The other components provide management functions, such as the QoS components for the virtual network operator and the infrastructure provider; they allow, among others, traffic differentiation within a virtual network and traffic differentiation between virtual networks. VIPER is composed of a client inside each virtual machine and a server in the Dom0 to support the plane separation paradigm.

VIPER extends the secure channel and the plane separation components presented in [8] to build a shared data plane with QoS support. Hence, VIPER ensures secure communication between DomU and Dom0 to exchange configuration and control data and offers two options for packet forwarding: plane separation, if the network requires packet forwarding with high performance, or packet forwarding through the virtual machine, if the network requires high flexibility.

IV. The resource sharing manager

VIPER resource sharing manager monitors the use of physical resources by each virtual network and punishes virtual networks that exceed the use of resources specified in the SLA. The control is based on observations in the short- and long-term uses of the CPU, memory, and bandwidth within Dom0.

The resource sharing manager assumes the existence of a specification of the physical resources allocated to each virtual network according to the SLA. This specification includes:

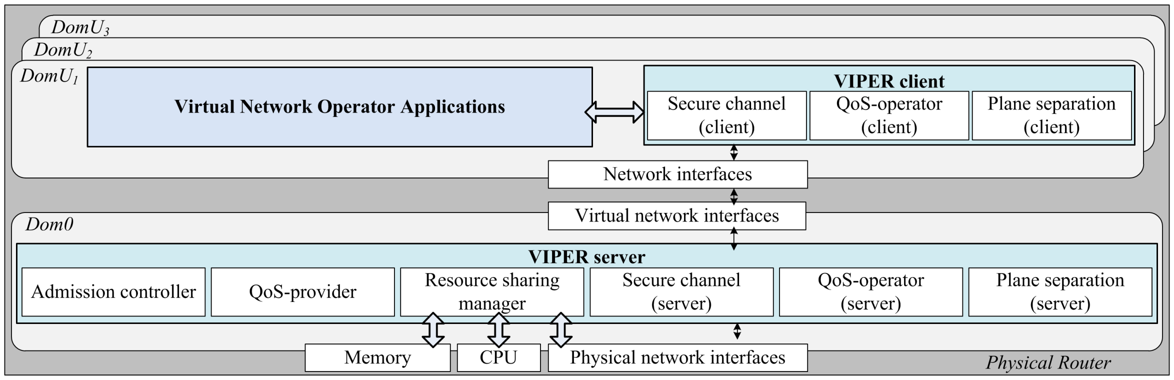


Fig. 2. The VIPER architecture comprises the resource sharing manager and the admission controller inside Dom0 and also provides plane separation with QoS support through a client-server model. VIPER also provides the QoS-operator component for differentiating the traffic among virtual networks.

- The *short-term rate reservation* ($R_s[n]$), which is the rate of resources that must be met for the virtual network n whenever there is demand in a short interval I_s.
- The *long-term volume reservation* ($V_l[n]$), which is an amount of resources $R_l[n] \cdot I_l$ that should be guaranteed during a long interval I_l, where $R_l[n]$ is the average rate of the long-term volume reservation of network n.
- The *short-term exclusive reservation* ($R_e[i]$), which is a type of short-term reserve that should not be made available to other networks, even if network i does not demand this resource.
- The *maximum volume reservation* ($V_{max}[i]$), which is the maximum resource usage that the network i can use during I_l.

Based on this set of features, a virtual network can define different profiles of physical resource usage. The choice of parameters for each virtual network varies according to the requirements of the virtual network operator and the price it is willing to pay for its slice of the network. A virtual network whose demand goes beyond the limits specified in the SLA is punished and sees its share of the resources reduced. In the following, we first show how to compute punishments and then how to apply them to the transgressive virtual networks.

A. Punishment calculation

The proposed resource sharing manager verifies whether the demand of network n, $D[n]$, is consistent with the SLAs. Algorithm 1 specifies how the resource sharing manager calculates the punishments for each network. [2] This algorithm runs in every interval I_s for each of the monitored resources of Dom0, i.e., the outgoing bandwidth of each physical link, the CPU, and the memory. It assumes that the demand $D[n]$ in the last I_s is known and this value is used as an estimation of network demand in the next I_s. Based on the estimated demand of all the networks, the resources are divided between the virtual networks. Algorithm 1 further assumes a prior knowledge of the number of virtual networks (N) hosted in the physical node, the total physical resources available to the virtual networks (R_t), the total resources used by each network up to the current time during the long interval ($used[n]$), and current weight of each virtual network n ($weight[n]$). The weight is a parameter generated by the resource sharing manager during each I_s to control the usage of the long-term volume reservation by each virtual network, so that the larger the weight, the greater the amount of resources besides the short-term rate reservation available to the virtual network in the next interval I_s.

Algorithm 1: Punishment calculation for each virtual network.

```
input : $R_t$, $N$, $I_s$, $I_l$, $R_s[\,]$, $V_l[\,]$, $weight[\,]$, $used[\,]$, $D[\,]$
output: $punish[\,]$
1  $Zero(punish[\,])$;
2  $total = R_t - \sum_{i=1}^{N}(R_s[i])$;
3  $total\_weight = \sum_{i=1}^{N} weight[i]$;
4  for $net = 1$ to $N$ do
5      if $(used[net] < V_l[net])$ then
6          $next[net] = R_s[net] + total \cdot weight[net]/total\_weight$;
7      else
8          $next[net] = R_s[net]$;
9      end
10     if $(D[net] > next[net])$ then
11         $punish[net] = 1 - next[net]/D[net]$;
12     end
13 end
14 while $(verify(next[\,], D[\,], N) == 1)$ do
15     $total\_weight = 0$;
16     for $net = 1$ to $N$ do
17         if $(next[net] > D[net])$ then
18             $next[net] = D[net]$;
19         end
20         if $((D[net] > next[net])\&(used[net] < V_l[net]))$ then
21             $total\_weight+ = weight[net]$;
22         end
23     end
24     $leftover = R_t - \sum_{i=1}^{N}(next[i])$;
25     for $net = 1$ to $N$ do
26         if $((next[net] < D[net])\&(used[net] < V_l[net]))$ then
27             $next[net]+ = leftover \cdot weight[net]/total\_weight$;
28         end
29         if $(next[net] < D[net])$ then
30             $punish[net] = 1 - next[net]/D[net]$;
31         else
32             $punish[net] = 0$;
33         end
34     end
35 end
```

In a more descriptive way, the algorithm works as follows. First, it calculates the amount of resources that are not used

[2]The extended version also considers the short-term exclusive reservation and the maximum amount of resources that should be provided to each virtual network.

by short-term rate reservations and sums up the weights of all networks. These values are used to calculate the amount of resources that must be released for each network in the next I_s, which is represented by the variable $next$ (line 6). All networks that have a demand higher than $next$ receive a punishment commensurate with the excess use (line 11).

To improve the resource distribution among virtual networks, the resource sharing manager verifies whether there is any network n with a demand $D[n]$ smaller than $next[n]$ (line 14). If there is at least one network in this condition, the amount of resources specified by the difference $next[n]-D[n]$ is redistributed to alleviate the networks that are being punished and have not exhausted the long-term volume reservation (line 27). The idea behind the redistribution of resources is to allow better network usage. Finally, the punishments of all networks are updated (lines 30 and 32). This process is repeated until the condition $D[n] < next[n]$ is false for all networks. Thus, the physical resources are distributed in proportion to the weight and the demand of each network.

B. *Weight update*

The manager stores a weight for each monitored resource of each virtual network. The weight of a network n is an adaptive variable, computed in each I_s based on the resources that the network n has used since the beginning of I_l ($used[n]$) and the long-term volume reservation ($V_l[n]$). The idea is to give to each network a weight proportional to the leftover long-term volume reservation. Thus, the networks that have more resources left (out of the long-term volume reservation), i.e. a greater $V_l[n] - used[n]$, gain priority in the allocation of available resources, receiving a larger slice of the resources if there is demand. Therefore, the resource sharing manager increases the likelihood that the long-term volume reservation is fully provided to all virtual networks.

C. *Punishing transgressive virtual networks*

The resource sharing manager computes the punishment as shown in Algorithm 1 for each monitored resource of each virtual network. The application of the computed punishment, $punish[\ ]$ (Alg. 1), however, varies according to the resource, i.e. bandwidth, CPU, or memory. When a virtual network n violates its processing resource reservations in Dom0, this network is punished by dropping a percentage $punish[n]$ of the packets on all incoming interfaces. Thus, as some packets fail to enter Dom0, it reduces the CPU power spent with packet forwarding. When there is bandwidth overload in an outgoing interface, packets destined to that interface are also dropped proportionately to $punish[n]$. It should be noted that punishment reduces network bandwidth consumption on the outgoing interface, but the CPU cost remains and it is accounted to the network CPU resources. Memory consumption is estimated by the amount of filtering rules and packet forwarding rules in Dom0 of each virtual network. Packet forwarding also demands memory in Dom0, but this demand is small and can be disregarded. The punishment due to excessive memory usage implies the disposal of a percentage $punish[n]$ of routes from the routing table. To prevent packet loss, a default route to the virtual machine is set up. Hence, if a packet has no forwarding rules set up in Dom0, this packet is forwarded by the data plane inside the virtual machine. Therefore, if a network uses the plane separation paradigm and depends on high performance in packet forwarding, it must control the number of routes and QoS rules installed in Dom0.

V. Admission control of new virtual routers

The virtual network admission controller arbitrates the access of new virtual routers to the physical machine. The number of virtual routers hosted in a physical machine influences the provision of long-term volume reservations. Indeed, the admission control mechanism is non-trivial because different network profiles may use the same amount of long-term resources. Then, the difference among network profiles, which include static information, such as disk and memory sizes, and dynamic information, such as throughput and CPU consumption, should be considered for calculating the resource blocking probability. The resource blocking probability is the probability of a virtual network service request, which is consistent with the SLAs, be denied due to an overload in the physical resource. The key idea of the proposed admission controller is to estimate the probability of blocking a resource demand. Our admission controller estimates the demand of the new virtual network and the variations in the resource demands of the hosted virtual networks before admitting a new virtual network. If the estimated blocking probability is greater than a threshold, then the new virtual network is not admitted.

A. *Monitoring and storage*

The proposed admission control stores a set of histograms representing the physical-substrate resource usage. A new histogram is generated for each dynamic resource in each monitoring period. A set of histograms of the same resource models the load variation along a period of time (e.g., a day).

Physical substrate histograms model the behavior of the aggregate resource usage of virtual networks. The idea is that networks with different traffic profiles lead to different aggregate resource histograms, even if the short and long term reservations are the same for all virtual networks. Thus, two networks can have the same V_l and R_c and have completely different behaviors, which imply in different aggregated resource demands. Thus, the physical substrate histogram realistically models what is happening with the physical network resources at each monitoring interval.

A monitoring interval is defined as $K_{adm} \cdot I_l$ seconds, where K_{adm} is an arbitrary constant chosen by the infrastructure administrator. To avoid a storage and processing overload, the proposed system randomly selects K_{rand}, which is the number of long intervals that will not be evaluated after K_{adm} long intervals. Moreover, instead of storing all histograms, the admission controller checks the difference between the current and the last histogram. For this, the admission controller normalizes both histograms and calculates the biggest error in the y-axis (e_m) between the two histograms. If condition $|e_m| > E_{adm}$ holds, where E_{adm} is a threshold specified by the infrastructure administrator, then the current profile is stored and a new histogram is started.

B. The admission control algorithm

The admission control algorithm estimates the blocking resource probability after a new virtual network joins the physical substrate. This estimated probability is used as a criterion for accepting or not a new virtual network. The algorithm inputs are the substrate histograms, the long-term volume reservation of each virtual network ($V_l[\,] = R_l[\,] \cdot I_l$), and the average resource usage of each virtual network, $R_{avg}[\,]$.

The admission control algorithm is accomplished in four steps:

- **Step 1 -** Estimate the aggregate resources usage through a histogram.
- **Step 2 -** Estimate how a demand increasing would impact in the aggregate resource usage (assuming that the reservations are respected).
- **Step 3 -** Estimate a probability function to model the new virtual network resource usage.
- **Step 4 -** Calculate the resource blocking probability if the new network were using the physical substrate.

After accomplishing these steps, the physical node uses the estimated resource blocking probability to decide whether to grant access to the new virtual network or not. In the following, we detail each step of the proposed algorithm.

1) Step 1 - monitoring: Step 1 consists of monitoring resource usage and storing histograms as described in Section V-A. Thus, in the end of this step, the algorithm knows the substrate histograms for bandwidth, CPU, and shared memory in each monitoring interval.

2) Step 2 - estimating a demand increase: In this step, the proposed algorithm estimates the substrate histograms if all virtual networks were using all reserved resources. The idea is to ensure that there will always be physical resources to meet all the virtual networks requirements, even if there are simultaneous peak demands. Solutions based on migration [2] are usually slow, because they depend on observing the resource blocking for a period of time, then searching for a new mapping between virtual and physical topology, avoiding overcharged physical nodes, and finally migrating the selected virtual nodes. Besides, these solutions can cause packet losses, leading to penalties for the infrastructure provider. Migration-based solutions without the use of an appropriate admission control may also overcharge physical nodes, causing more losses and oscillations in the mapping of virtual networks over the physical substrate. Therefore, it is important to estimate the impact of a new network before admitting it into a physical node.

VIPER estimates the resource blocking probability as if the new virtual network was using the physical network and all hosted networks were fully using their reservations. Hence, the proposal estimates how the histogram that stores the resource usage in the physical substrate would look like if all the virtual networks were using all the reserved resources. The proposed algorithm uses the average resource usage, $R_{avg}[\,]$, and the average long-term volume reservation, $R_l[\,]$, of each monitored resource to estimate the variation between the current demand and a increased demand, which is represented by

$$\Delta = \sum_{n=1}^{N_H} R_l[n] - R_{avg}[n], \tag{1}$$

where N_H is the number of virtual networks. The Δ estimates the impact of a demand increase over the physical substrate. As a basis for this estimate, we assume that virtual networks will request all reserved resources and that the demand increase does not change the way resources are required. Thus, if the function $f_i(t)$ models the resource usage of virtual network i over time, then the estimate of the increased resource usage of this network would be given by $f_i(t) + \Delta_i$, where $\Delta_i = R_l[i] - R_{avg}[i]$. Nevertheless, it is irrelevant for the proposed algorithm the way each network increases its consumption individually, but the way the aggregated usage increases when each virtual network is using its whole reservation. After obtaining Δ (Eq. 1), the controller updates the substrate histograms for bandwidth, CPU, and memory. First, each histogram is shifted according to Δ. Assuming that $I[i]$ is an histogram interval with upper bound $L_s[i]$ and $I[i']$ is an interval that contains the value $L_s[i] + \Delta$, then

$$L_s[i' - 1] < L_s[i] + \Delta \leq L_s[i']. \tag{2}$$

Furthermore, we assume that $H_{sub}(I[i])$ is the number of event occurrences in the interval $I[i]$ in the substrate histogram and that N_{int} is the number of intervals in substrate histogram. Thus, the shifted substrate histogram, H_{sft}, is computed as $H_{sft}(I[i']) = H_{sub}(I[i])$. The interval $I[N_{int}]$ of H_{sft} is defined as

$$L_s[N_{int} - 1] < I[N_{int}] < \infty, \tag{3}$$

to maintain a fixed number of intervals. Figure 3 shows an example of histogram shifting when $N_{int} = 10$ and $\Delta = 1$.

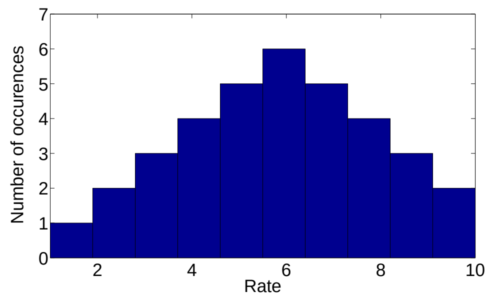
After that, the controller calculates a probability mass function for the shifted substrate histogram (PMF_{hist}), assuming that the values of the x-axis are given by $L_s[s]$, as shown in Figure 3(c).

3) Step 3 - estimating the demand of the new virtual network: The controller estimates the demand of the new virtual network based on a predefined distribution. For example, an optimistic admission controller assumes a Poisson distribution, while a pessimistic admission controller assumes a distribution concentrated at peak rates.

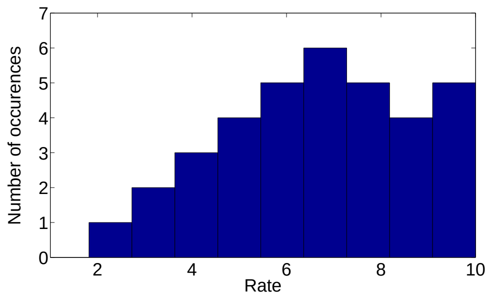
The distribution estimated for the new virtual network (PMF_{new}) is represented according to the intervals $I[n]$ of the substrate histogram, assuming that the values of x-axis are given by $L_s[\,]$.

4) Step 4 - estimating the resource blocking probability: The controller algorithm estimates the resource blocking probability supposing that the new virtual network joins the substrate. The probability is calculated for each dynamic resource, i.e. CPU, bandwidth, and memory, and is used as criteria to accept or not the new virtual network.

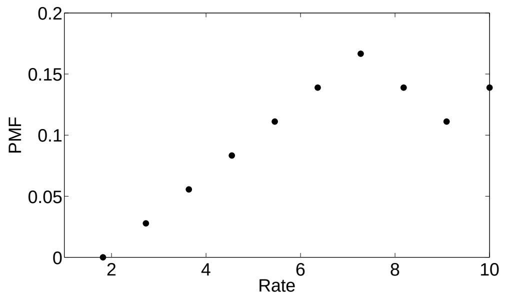
The proposed algorithm uses PMF_{hist}, which is based on the shifted substrate histogram, to calculate the resource blocking probability, according to the distribution assumed for



(a) Substrate histogram (H_{sub}).



(b) Shifted substrate histogram (H_{sft}).



(c) Probability mass function of the shifted substrate histogram.

Fig. 3. Example of substrate histogram shifting, when $N_{int} = 10$ and $\Delta = 1$.

the new virtual network (PMF_{new}). Considering that

$$A = \{(x_1, x_2) \mid x_1 + x_2 \geq C\}, \\ \text{where } x_1, x_2 \in \{L[1], L[2], \ldots, L[N_{int}]\}, \tag{4}$$

is a set of tuples (x_1, x_2) whose sum $x_1 + x_2$ exceeds the resource capacity C, then the controller computes the resource blocking probability by

$$P_B = \sum_{\forall (x_1, x_2) \in A} PDF_{hist}(x_1) \cdot PDF_{new}(x_2). \tag{5}$$

Therefore, the proposed algorithm estimates the probability of the sum of the resources used in the shifted substrate histogram and in the new network exceeds the physical machine capacity.

The new virtual router is accepted if there are enough resources for the static requirements and if the conditions

$$\sum_{n=1}^{N} R_s[n] < C \qquad \text{and} \qquad P_B < P_L \tag{6}$$

hold for all histograms of all dynamic resources, where P_L is the resource blocking probability threshold accepted by the infrastructure administrator. A small P_L guarantees a low probability of packet losses. Nevertheless, it also reduces the physical resource efficiency, which reduces the infrastructure provider profits.

VI. QoS Provision

One of the main advantages of VIPER is the support for QoS provisioning. Although a virtual router that does not use the plane separation paradigm can set up a traffic control inside its virtual machine, a control at this level is not enough to ensure quality of service. A virtual router has no control over its own traffic while it is being forwarded by Dom0 from the physical device driver to the virtual device driver, which may include delays and packet losses. Thus, VIPER offers primitives for adding QoS rules inside a virtual network and among virtual networks, as shown in Fig. 4 and 5, for packet forwarding through the virtual machine and through Dom0, respectively. According to this scheme, the infrastructure provider can configure priority access for the physical hardware to some virtual network as well as any virtual network operator can differentiate its own packets being processed inside the virtual machine.

The use of the plane separation paradigm implies that the data plane of each virtual network, which was previously inside DomU, is placed into Dom0. Thus, the QoS provision must be adapted to work with the plane separation paradigm. As a consequence, the QoS provision for both the virtual network operator and the infrastructure provider must be supplied inside Dom0, as shown in Fig. 5. VIPER provides an interface to each virtual router configure its own QoS rules through the QoS-operator client and QoS-operator server components. These components guarantee that one virtual network can configure its QoS rules without interfering with the QoS rules of other virtual networks.

VII. Simulation results

The virtual network admission control for the proposed architecture was implemented in C++. The proposed algorithm is evaluated under different traffic patterns through simulations. We compare our proposal to other proposals of the literature, namely Sandpiper [2] and VNE-AC (Virtual Network Embedding Algorithm based on Ant Colony Metaheuristic) [13].

Sandpiper is a system for monitoring virtual machines load in data centers. When a physical machine is overloaded, which is identified when requests for resources of virtual machines are blocked, Sandpiper looks for a new virtual machine that is able to host one or more virtual servers of the overloaded node. The admission control in Sandpiper is based on the current virtual machine peak load and on the average amount of idle resources in the new physical machine. If the resource demand peak, which is estimated by the 95% rate in the cumulative distribution function, is less or equal to the average amount of idle resources, then the new virtual network is admitted.

VNE-AC is a virtual network mapping mechanism. This algorithm is based on the ant colony metaheuristic, which determines an adequate mapping of virtual networks over the physical substrate. The admission control mechanism proposed

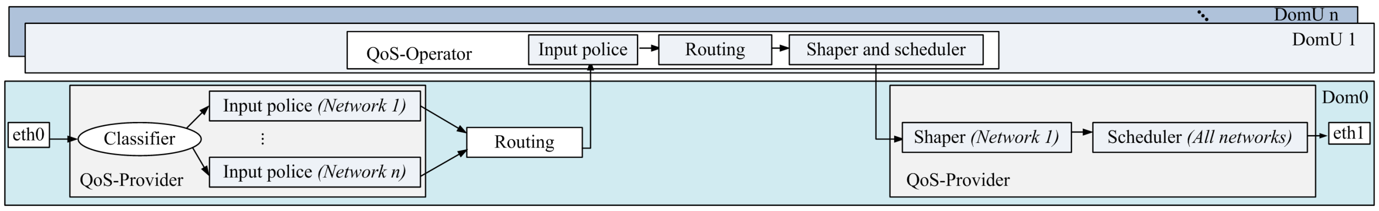


Fig. 4. QoS provision in VIPER, assuming packet forwarding through the virtual machine.

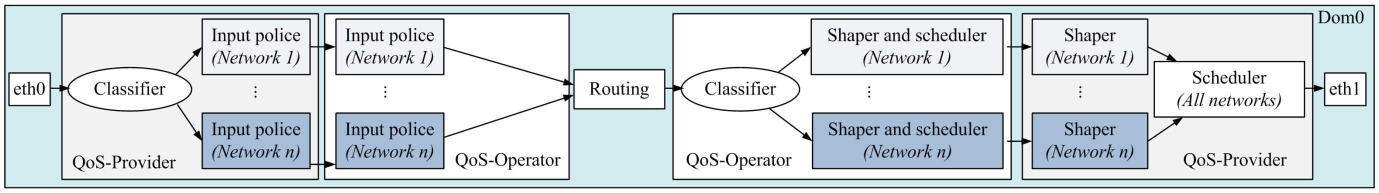


Fig. 5. QoS provision in VIPER, assuming the usage of the plane separation paradigm, in which the packet forwarding is performed in Dom0.

in the VNE-AC assumes that resources are statically attributed to each virtual network. Hence, this admission control restricts the SLA specification for each virtual network.

We implemented both Sandpiper and VNE-AC admission control to compare them to our mechanism. The Sandpiper peak rate is chosen as $p_k = 95\%$, as suggested by Sandpiper's authors. VNE-AC reservation threshold is set as R_l, which is the average rate of the long-term volume reservation (V_l). We also evaluate the impact of Δ (Eq. 1). We compare our proposal to a simpler version that does not shift the histogram, which is indicated as 'Hist' in the results, to verify the efficiency of this procedure.

In this evaluation, we measure the number of virtual networks that each mechanism admits to be hosted in a physical router. We assume, for simplicity, that all virtual networks present the same resource reservation parameters and that the number of intervals in the histogram for our proposal is $N_{int} = 30$.

For evaluating the results, we also measured the ideal number of virtual networks in the analyzed physical node in each experiment. This ideal number is chosen as the maximum number of virtual networks that guarantees that the blocking probability threshold specified by the infrastructure administrator (P_L) is not violated. Since $p_k = 95\%$ for Sandpiper, we selected $P_L = 0.05$ for fairness.

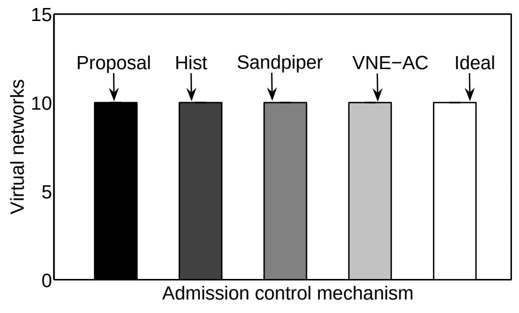
In one hand, if an admission control mechanism admits less networks than the ideal number, than the physical machine resources are wasted. On the other hand, if the admission control mechanism admits more virtual networks than the ideal number, then the physical node is overloaded and the reserved resources are denied for the virtual networks, generating fines for the infrastructure provider. Hence, admitting more virtual networks than the ideal value is worse than admitting less virtual networks.

We run 30 rounds for each experiment, analyzing the output link throughput in packets/s. We present in the results the corresponding mean value and the standard deviation.

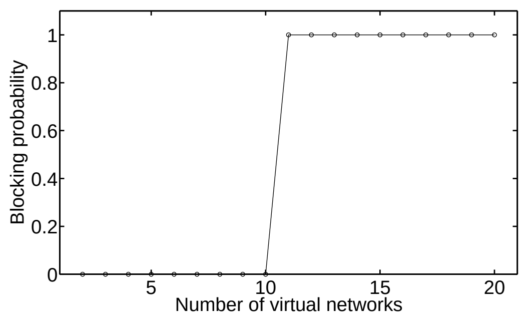
A. *Traffic pattern impact*

In the first experiment, we evaluate virtual networks whose traffic is modeled by Poisson processes. Each virtual network demands ≈ 100 Mb/s, which is also the value of R_l. Figure 6(a) shows that all mechanisms admitted the ideal number of virtual networks. Hence, if the virtual network traffic presents a small deviation, then all the analyzed mechanisms are able to correctly perform the admission control. Figure 6(b) shows that the admission control in this scenario is abrupt, because ten networks cause no blocking probability, while eleven networks cause a 100% blocking probability.

To evaluate the proposals in environments with a higher variability, we also simulated virtual networks with a traffic pattern described by an on-off model. These results are on Figure 7. In this experiment, all virtual networks present the same traffic pattern. The new virtual network traffic pattern is estimated based on a Poisson process in the proposed mechanism, but it behaves just as all the other virtual networks, with an on-off traffic. The on-off traffic is generated based on an exponential distribution with $\mu = 1/3$. Each value generated with the exponential distribution indicates a time interval in which the traffic will be on or off. The on-traffic is modeled by a Poisson distribution with $\lambda \approx 200$ Mb/s and the off-traffic is always zero. Figure 7(a) shows that Sandpiper and VNE-AC admits more virtual networks than the other proposals. Nevertheless, both mechanisms admit more virtual networks than the ideal number of virtual networks. By Figure 7(b), we observe that the blocking probability for Sandpiper and VNE-AC, which admitted 10 virtual networks, is about 50%. Sandpiper overestimated the idle resources due to the on-off nature of traffic and admitted more networks than the ideal. VNE-AC behaves similarly to Sandpiper, but for different reasons. VNE-AC does not differentiate the scenario of Figures 6 and 7, because real data is not used for predicting the demand. Hence, virtual networks with different data patterns are equally treated, generating admission control errors. Our proposal admitted the ideal number of networks, guaranteeing a low blocking probability and an efficient resource usage.

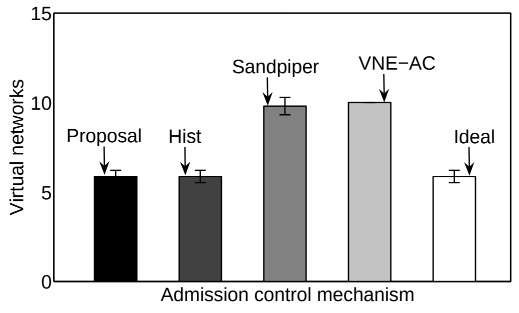


(a) Number of virtual network admitted by each mechanism.

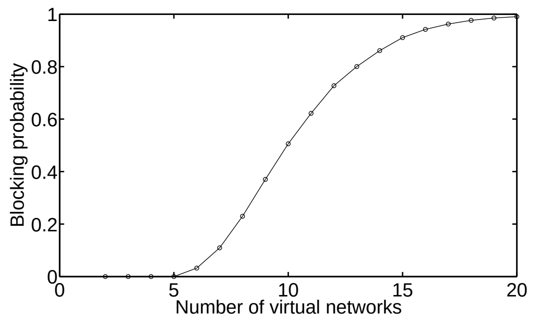


(b) Blocking probability according to the number of virtual networks in parallel.

Fig. 6. Admission control assuming virtual networks with traffic modeled by a Poisson process and maximum blocking probability of 5%.



(a) Number of virtual network admitted by each mechanism.



(b) Blocking probability according to the number of virtual networks in parallel.

Fig. 7. Admission control assuming virtual networks with on-off traffic and maximum blocking probability of 5%.

B. Δ *Impact*

This experiment evaluates virtual networks whose demand varies with time. In this scenario, the throughput of the virtual networks increases with time up to the threshold of the long-term volume reservation. The admission request for the new virtual network is sent when the demand of the other virtual networks correspond to half of their reservation. Figure 8(a) shows that Hist and Sandpiper accepted all the analyzed networks, because these mechanisms do not consider the demand increase. Thus, the blocking probability is of 100% for both mechanisms when the demands increase up to their reservation threshold in the end of the simulation, as shown in Figure 8(b). Both our proposal and VNE-AC consider the demand variation and achieved a number of accepted virtual networks close to the ideal number.

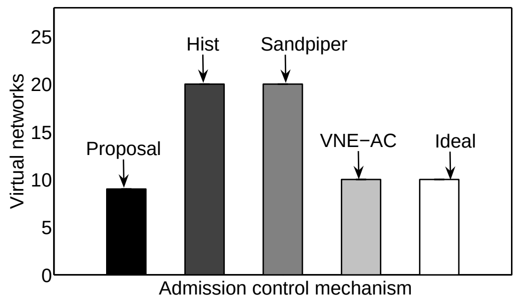
Based on these results, the proposed admission control is the only one that is efficient in all analyzed scenarios, guaranteeing the admission of a high number of virtual networks, without violating the blocking probability threshold imposed by the infrastructure administrator. Sandpiper and VNE-AC are inefficient when network demand presents a high variability, overcharging the physical node.

VIII. Experimental evaluation

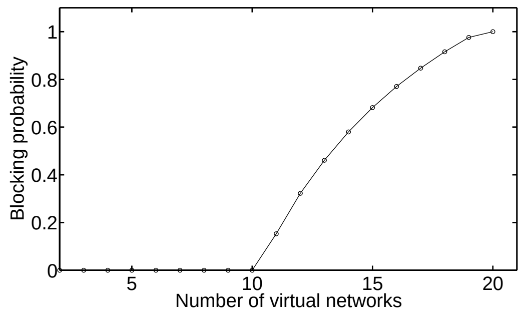
To evaluate VIPER in a real setup, we developed a prototype in C++ combined with Xen-4.0 configured in router mode. CPU usage, which is estimated based on the volume of forwarded packets, and bandwidth usage are monitored using the Iptables tool. We also use Iptables to apply punishments. To control the QoS parameters, we used the Traffic Control (TC) tool. The testbed is composed of four machines, out of which one runs Xen and VIPER, representing the physical router. The remaining three, called external machines, are directly connected to this physical router. The physical router is a machine with an Intel Core 2 Quad, 4GB RAM, and five-gigabit network interfaces, hosting three virtual machines with the operating system Debian 2.6.32-5. The Dom0 is configured with four logical CPUs, while each DomU has a logical CPU. The traffic is generated by the Linux kernel module ‘pktgen’, characterized by UDP packets with 1,472-byte payloads. The parameters of the resource sharing manager are $I_s = 1$ s (for the short interval) and $I_l = 15$ s (for the long interval).

A. *Resource sharing for different reservation patterns*

We analyzed VIPER controlling three virtual networks with different reservations and equal demands given by $D[n] = 300$ Mb/s, $0 < n < 3$. Network 1 has a short-term rate reservation $R_s[1] = 50$ Mb/s and an average rate of long-term volume reservation $R_l[1] = 350$ Mb/s, so that its demand is consistent with SLAs and must be fully provided ($D[1] < R_l[1]$). Network 2 has $R_s[2] = R_l[2] = 100$ Mb/s, which means that this network requires more than the resources



(a) Number of virtual network admitted by each mechanism.



(b) Blocking probability according to the number of virtual networks in parallel.

Fig. 8. Admission control assuming virtual networks with increasing traffic and maximum blocking probability of 5%.

agreed in the SLAs ($D[2] > R_l[2]$). Network 3 simulates a network with high traffic rate, but without any priority or delay requirement, so that $R_s[3] = 0$ and $R_l[3] = 250$ Mb/s. Therefore, Network 3 also presents a demand that exceeds its SLA ($D[3] > R_l[3]$). CPU resources and memory are equally divided among the three networks. The traffics of Networks 1 and 2 are transmitted from an external machine to another via Dom0 routing, because these networks use the plane separation paradigm. The traffic of Network 3 goes from the virtual router to an external machine. The output link is shared by the three traffics.

Fig. 9(a) shows the throughput $\phi[n]$ of the three networks over time in the output link. Because Network 2 has only the short-term rate reservation, the resource sharing manager limits its use steadily at 100 Mb/s. Network 1 has its demand met at all times, while Network 3 losses more packets when the total amount of data available in the long-term volume reservation has been used.

Fig. 9(b) shows the evolution of the adaptive weight for each virtual network. The weight of Network 2 is always zero, because this network has no right to use anything beyond the short-term rate reservation. The weight of Network 1 increases while the weight of Network 3 decreases, because Network 3 consumes faster its long-term volume reservation. The weight fits proportionately between the networks according to the slice of long-term volume reservation that each network can still use.

Finally, we computed the short-term error, the long-term error, and the compliance. These parameters check if the SLAs of each of the virtual network are respected. The short-term error of Network n, $E_s[n]$, is computed at each short interval and demonstrates to which level the provision of the short-term rate reservation was not met. Thus,

$$E_s[n] = \max\left(0, 1 - \frac{\phi[n]}{\min(R_s[n], D[n])}\right). \qquad (7)$$

Similarly, the long-term error of Network n, $E_l[n]$, is computed at the end of long intervals and shows to which level the provision of the long-term volume reservation was disrespected. The compliance of network n, $C_p[n]$, is computed after I_l and shows an average level to which the SLA was respected. Then,

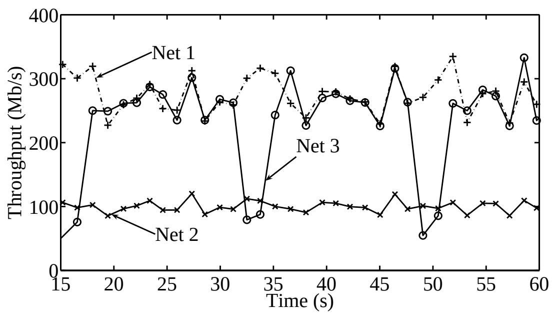
$$C_p[n] = 1 - \frac{\text{mean}(E_s[n])}{2} - \frac{E_l[n]}{2}. \qquad (8)$$

The results show that all networks obtained $C_p = 1$ in all tests, because the resource sharing manager offers an efficient control of the shared resources. Hence, our resource manager is able to correctly ensure both the short-term rate reservation and the long-term volume reservation to all networks in this scenario.

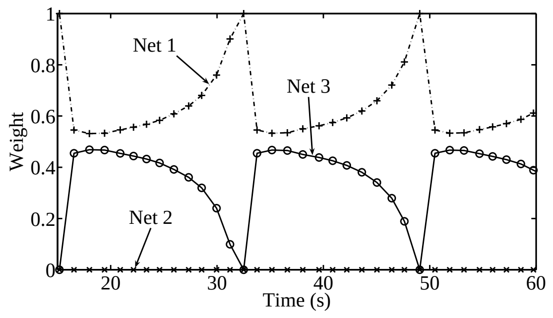
B. Long-term volume reservation provision

In the second test, we set up a scenario with two virtual networks, one using plane separation (Network 1) and other forwarding packets through the virtual machine (Network 2). The networks operate at $R_s[1] = R_s[2] = 100$ Mb/s, $R_l[1] = 600$ Mb/s and $R_l[2] = 200$ Mb/s. Both networks request more bandwidth than specified in the SLA, because $D_1 = D_2 = 1$ Gb/s. We set up a flow for each virtual network between two external machines and both flows share the output link.

In this test, VIPER is compared to XNetMon (XMon in the plots) [8], because both proposals control the resource sharing between virtual networks. XNetMon, however, does not limit resource usage while there are available resources. We also compare VIPER with three different traffic control profiles created with the Traffic Control (TC) tool. We used the Hierarchical Token Bucket (HTB) queuing discipline to simulate behaviors similar to VIPER. The first profile, called TC1, provides a minimal reservation $R_m[n] = R_l[n]$ and allows traffic to reach up to the link capacity (1 Gb/s) if there are demand and available resources. The profile TC2 provides $R_m[n] = R_s[n]$, but limits the link usage at the average rate of the long-term volume reservation, $R_l[n]$. Hence, TC2 presents the closest behavior to VIPER. The profile TC3 provides $R_m[n] = R_s[n]$ and allows a traffic to consume all the resources, if they are available, being closest to the behavior of XNetMon. Although TC is a precise tool for traffic control, it does not control shared resources like memory and CPU, such as VIPER and XNetMon do. Thus, to make the comparison fair, both VIPER and XNetMon do not make restrictions on the use of CPU and memory for any of the networks. Besides, we also analyze the results when no control is applied to the virtual networks (w/o in the plots).

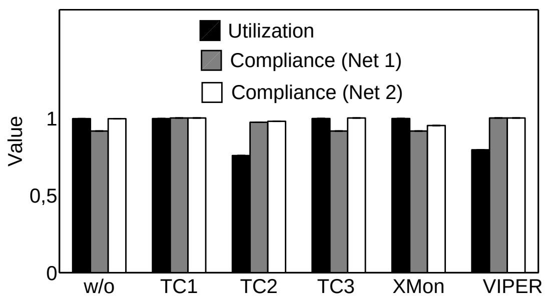


(a) Dom0 throughput per VN.

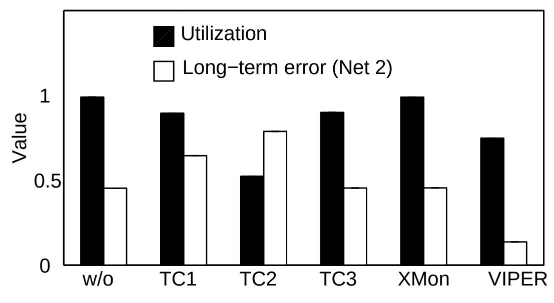


(b) Weight variation per VN.

Fig. 9. Resource sharing control according to different virtual networks (VN) patterns, assuming demand of 300 Mb/s per network.



(a) Compliance assuming two VNs with equal demand but different R_l.



(b) Long-term error of Network 2 assuming a burst demand in this VN.

Fig. 10. Compliance, long-term error, and short-term error using different control tools, assuming different virtual networks (VN) profiles.

Fig. 10(a) presents the results for utilization, which is the ratio between the throughput and the link capacity, and compliance, both with respect to the use of the output link. Only VIPER and profiles TC1 and TC2 provide the maximum compliance to both networks. In the other profiles, Network 2 was privileged due to the low ability to differentiate the use of resources beyond the minimum reservation of each network. Profiles TC1 and TC2 present a good compliance because they are adapted to differentiate traffic also based on the rate of the long-term volume reservation. Nevertheless, only VIPER and TC2 show low link utilization, which is a must-have characteristic in virtualized networks.

The next test is based on the same network control profiles. In this test, however, we change the network demands: the demand of Network 1 is $D[1] = 1$ Gb/s during the whole test, while the demand of Network 2 is $D[2] = 0$ during the first two thirds of the long interval I_l and $D[2] = 1$ Gb/s for 3 s during the last third of I_l. This scenario simulates the behavior of a network with burst traffic, in which the burst is equal to the total long-term volume reservation of Network 2. Fig. 10(b) shows that VIPER presents the lowest long-term error, because of the adaptive weight. The long-term error of VIPER is more than five times smaller than the long-term error of profile TC2, which presented a similar result to VIPER in the previous test. Furthermore, VIPER provides the second lowest utilization. Hence, VIPER is an advantageous solution for both the virtual network operator, which wants a resource provision as agreed in the SLAs, and the infrastructure provider, which requires a low use of physical resources to allocate more networks over the same physical hardware.

C. QoS provision

Finally, we evaluate the QoS components of VIPER. Fig. 11 shows the impact of the use of QoS premises by the infrastructure provider. In this scenario, composed of two virtual networks, it is assumed that Network 1 hires priority for its packets to guarantee a small forwarding delay. It is also assumed the following parameters in the output link: $R_s[1] = 50$ Mb/s and $R_l[1] = 400$ Mb/s for Network 1 and $R_s[2] = 100$ Mb/s and $R_l[2] = 600$ Mb/s for Network 2. CPU and memory resources are equally divided between the networks. Network input demands are, respectively, $D[1] = 50$ Mb/s and $D[2] = 1$ Gb/s. We chose a high $D[2]$, because a high volume of traffic hinders the provision of QoS for Network 1. The traffic of Network 2, which uses the plane separation paradigm, flows among two external machines. The traffic of Network 1, which is routed through the virtual machine, is generated by a different external machine and shares the output link with the traffic of Network 2.

Fig. 11 shows the results for the Round Trip Time (RTT) of both networks with and without VIPER. In the first scenario, called ‘w/o prio’, both networks have the same priority and, in the second scenario, called ‘prio’, the traffic of Network 1 is privileged. When we give priority to one virtual network, even

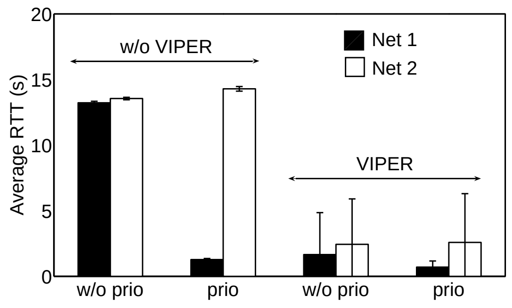


Fig. 11. RTT according to the QoS parameters inter VNs, assuming that Net 1 is prioritized.

without the use of VIPER resource sharing manager, the RTT decreases by more than 10 times for the privileged traffic. The use of VIPER ensures that Network 2 does not exceed the use of network or CPU, reducing the volume of processed data and thereby further reducing the transmission delay. Hence, VIPER QoS component reduces the RTT by more than 18 times when compared to the scenario without VIPER and without priority and more than 1.8 times when compared to the scenario with a privileged traffic but without the resource sharing manager of VIPER.

IX. Related Work

Most of works concerning virtual resource management focus on data centers and cloud systems. Gong et al. proposed a system called PRedictive Elastic reSource Scaling (PRESS), which predicts application resource demands using Markov chains and Fast Fourier Transforms [14]. The predictions are used to automatically adjust the resource allocation, according to the service level objectives. The system was implemented over Xen and tested under different server application loads. Another resource sharing management technique is the virtual machine migration, which is used in Sandpiper [2]. Sandpiper is a centralized control system that detects hotspots, determines a new mapping of physical to virtual resources, and migrates virtual machines. Although proposals such as PRESS and Sandpiper focus on the resource provisioning for virtual machines, these approaches deal only with datacenter loads, which are different from the virtual networking resource requirements. Virtual router demands are mainly determined by I/O operations, which imply in different resource consumption profiles. Besides, many virtualization techniques present isolation failures under a high I/O load [3], [8], but this challenge is not addresses by resource management control systems for data centers. VIPER is a system specialized in virtual networking, which guarantees virtual router isolation even under high traffic loads.

Concerning isolation in virtual networks, Egi et al. investigate the construction of a platform of virtual routers using Xen and Click [15], evaluating the provision of isolation and fairness between the networks [5]. The authors investigate the use of different data planes, assuming routing through Dom0 and through a virtual machine, and evaluate the ability to share resources among virtual networks. This work, however, has no mechanisms for defining management capabilities to specify and control the amount of resources of each virtual network. Also, the authors' proposal does not differentiate or prioritize traffic to ensure QoS in the virtual networks. Another approach based on Linux and Click to create a shared data plane is proposed by Keller and Green [16]. In this proposal, each virtual network can create its own data plane, based on generic packet forwarding primitives defined in Click. The authors, however, do not address a fair division of resources among the control planes. Hence, these approaches and VIPER are complementary, allowing the provision of a flexible data plane while still guaranteeing isolation and QoS.

Xen Network Monitor (XNetMon) is a monitor for Xen networks that guarantees the virtual network isolation and avoids that malicious virtual networks interfere in the other virtual network performance [8]. Although this system differentiates resources among networks, the resource allocation parameters are inflexible. XNetMon main goal is to prevent malicious virtual routers from harming other networks and it does not focus on the developing a management interface or a virtual network admission control, neither the provision of QoS guarantees for virtual networks, such as in VIPER.

Trellis [3] is a high-performance packet forwarding system that achieves isolation in the virtual links on operating-system level virtualization platforms. Trellis implements a custom high-performance bridge, besides implementing virtual links using the proposed Ethernet-over-GRE (EGRE) tunnels. Trellis enforces that the bandwidth and the jitter of a virtual network do not influence other virtual networks. Nevertheless, Trellis does not treat isolation problems during packet forwarding, such as VIPER. Trellis neither controls the admission of new virtual networks.

OpenFlow [17] is a platform with support for network virtualization that is based on a centralized control plane. In OpenFlow networks, the virtualization is controlled by a special node that works as proxy between the network controllers and the OpenFlow switches, called FlowVisor [18]. FlowVisor creates slices in OpenFlow networks by controlling CPU, memory, and bandwidth usage by each virtual network. Nevertheless, FlowVisor provides a SLA definition more restrictive than VIPER and also does not perform virtual network admission control.

Another important issue in the management of virtual networks is the admission control of new virtual routers [2], [13]. Fajjari et al. proposes a virtual network mapping based on an ant colony metaheuristic [13]. This proposal increases the number of accepted virtual networks by remapping the hosted virtual networks to provide a slice in the physical substrate with enough resources to host the new virtual network. This proposal, however, statically maps the resource requirements in each virtual router, which may cause an under-provisioning for the virtual networks. Wood et al. also proposes an admission control for virtual machines, as described in Section VII, which verifies whether the estimated peak demand is lower than the idle resources in the physical substrate [2]. This proposal, however, is not as profitable as VIPER, as shown by the results.

VIPER presents a precise and efficient resource sharing manager, which considers flexible SLAs to describe each virtual network traffic requirements. Besides, our proposal provides QoS control for both the infrastructure provider and the virtual network operator, reducing delays in prioritized networks. VIPER also offers a management interface to define and control the resources of each virtual network, besides assessing new virtual network hosting on the physical substrate.

X. Conclusion

Virtualization systems such as Xen still present serious vulnerabilities when used for network virtualization, because of the challenges to provide isolation in I/O operations. In this paper, we proposed VIPER, an efficient QoS-aware system for control and management of virtual networks. VIPER ensures strong isolation between networks and the compliance with contracted SLAs between the infrastructure provider and the virtual network operator. The main contributions of VIPER are the resource sharing manager, which guarantees the right division of resources between virtual networks, the components to provide QoS, enabling the inclusion of QoS parameters at the level of infrastructure provider, and the admission control for new virtual routers, which guarantees that there are enough resources to attend the SLAs of all hosted virtual networks. These features significantly differentiate VIPER of other resource control systems in the literature.

The characteristics used to model the SLAs in VIPER, which include the short and long-term reservations, and the setting of QoS parameters, assure the modeling of an extensive set of network configurations. VIPER provides high manageability of network resources in virtualized environments while solving the existing problems in the Xen platform. Through extensive simulation and experimental analyses, we show that VIPER guarantees high probability of providing all the reserved resources and still allow efficient resource usage by a precise control of the access of new virtual routers. Furthermore, the tests with the developed prototype show that VIPER guarantees full compliance between the forwarded traffic and the SLAs, besides guaranteeing small delays for privileged traffic.

Acknowledgment

The authors would like to thank FINEP, FUNTTEL, FAPERJ, CNPq, CAPES, and FUJB.

References

[1] N. Feamster, L. Gao, and J. Rexford, “How to lease the Internet in your spare time,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61–64, Jan. 2007.

[2] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, “Black-box and gray-box strategies for virtual machine migration,” in *Proceedings of the 4th USENIX conference on Networked systems design & implementation (NSDI’07)*. USENIX Association, 2007, pp. 229–242.

[3] S. Bhatia, M. Motiwala, W. Muhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, “Trellis: A platform for building flexible, fast virtual networks on commodity hardware,” in *Proceedings of the Workshop on Real Overlays and Distributed Systems (ROADS)*, Dec. 2008, pp. 1–6.

[4] N. C. Fernandes, M. D. D. Moreira, I. M. Moraes, L. H. G. Ferraz, R. S. Couto, H. E. T. Carvalho, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, “Virtual networks: Isolation, performance, and trends,” *Annals of Telecommunications*, pp. 1–17, 2010.

[5] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, “Evaluating Xen for router virtualization,” in *International Conference on Computer Communications and Networks (ICCCN’07)*, Aug. 2007, pp. 1256–1261.

[6] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, “Fairness issues in software virtual routers,” in *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow (PRESTO ’10)*, Aug. 2008, pp. 33–38.

[7] W. Almesberger, “Linux network traffic control - Implementation overview,” in *5th Annual Linux Expo*, 1999, pp. 153–164.

[8] N. C. Fernandes and O. C. M. B. Duarte, “XNetMon: A network monitor for securing virtual networks,” in *ICC 2011 Next Generation Networking and Internet Symposium (ICC’11 NGNI)*. IEEE, Jun. 2011, pp. 1–6.

[9] C. A. Waldspurger, “Memory resource management in VMware ESX server,” *SIGOPS Operating System Review*, vol. 36, pp. 181–194, December 2002.

[10] Y. Wang, E. Keller, B. Biskeborn, J. V. der Merwe, and J. Rexford, “Virtual routers on the move: Live router migration as a network-management primitive,” in *ACM SIGCOMM*, Aug. 2008, pp. 231–242.

[11] B. des Ligneris, “Virtualization of linux based computers: The linux-vserver project,” *High Performance Computing Systems and Applications, Annual International Symposium on*, vol. 0, pp. 340–346, 2005.

[12] J. Carapinha and J. Jimenez, “Network virtualization - a view from the bottom,” in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures (VISA ’09)*. ACM, 2009, pp. 73–80.

[13] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, “VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic,” in *ICC 2011 Next Generation Networking and Internet Symposium (ICC’11 NGNI)*. IEEE, Jun. 2011, pp. 1–6.

[14] Z. Gong, X. Gu, and J. Wilkes, “PRESS: Predictive elastic resource scaling for cloud systems,” in *6th International Conference on Network and Services Management (CNSM 2010)*. IEEE, Oct. 2010, pp. 9–16.

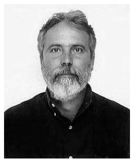
[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, “The click modular router,” *Operating Systems Review*, vol. 5, no. 34, pp. 217–231, Dec. 1999.

[16] E. Keller and E. Green, “Virtualizing the data plane through source code merging,” in *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow (PRESTO ’08)*, Aug. 2008, pp. 9–14.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S., and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[18] R. Sherwood, M. Chan *et al.*, “Carving research slices out of your production networks with OpenFlow,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 129–130, 2010.

Natalia Castro Fernandes received an Electronics and Computer Engineering degree in 2006, a M.Sc. degree in 2008, and a D.Sc. degree in 2011 from Federal University of Rio de Janeiro, Brazil. Currently, she is a professor with Fluminense Federal University. Her major research interests are in virtual networks, security, and Future Internet.



Otto Carlos M. B. Duarte received the electronic engineer degree and the M.Sc. degree in electrical engineering from UFRJ, Brazil, in 1976 and 1981, respectively, and the Dr. Ing. degree from ENST/Paris, France, in 1985. Since 1978, he has been a professor with UFRJ. His major research interests are in security, and mobile communications.