Analysis and evaluation of a XEN based virtual router

Fabienne Anhalt, Pascale Primet

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Analysis and evaluation of a XEN based virtual router

Fabienne Anhalt — Pascale Vicat-Blanc Primet

N° ????

Septembre 2008

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

Analysis and evaluation of a XEN based virtual router

Fabienne Anhalt , Pascale Vicat-Blanc Primet

Thème NUM — Systèmes numériques
Projets ANR CIS 06 HIPCAL 005

Rapport de recherche n° ???? — Septembre 2008 — 60 pages

Abstract: Virtualization techniques are applied to improve features like isolation, security, mobility and dynamic reconfiguration in distributed systems. To introduce these advantages into the network where they are highly required, an interesting approach is to virtualize the internet routers themselves. This technique could enable several virtual networks of different types, owners and protocols to coexist inside one physical network. In this study, we propose a model of a virtual router we have implemented with XEN and we evaluate its properties. We show that the performance is close to the performance of non virtualized software routers, but causes an important processing overhead and unfairness in the share of the ressources. We study the impact of the virtual machine scheduler parameters on the network performance and we show that the module which is responsible of forwarding the packets between the virtual machines and the physical interfaces is the critical point of network communications.

Key-words: Network performance, virtualization, XEN, scheduling

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Analyse et évaluation d'un routeur virtuel basé sur XEN

Résumé : La virtualisation est une technique employée pour améliorer des facteurs comme l'isolation, la sécurité, la mobilité et la reconfiguration dynamique des systèmes distribués. Afin d'introduire ces avantages dans les réseaux où ils sont fortement requis, une approche intéressante est de virtualiser les routeurs de l'Internet lui-même. Cette technique permettrait de faire cohabiter plusieurs réseaux virtuels de types, de propriétaires et de protocoles différents au sein d'un même réseau physique. Dans cette étude, nous proposons un modèle de routeur virtuel que nous avons implémenté à l'aide de XEN et nous évaluons ses propriétés. Nous montrons qu'une performance proche de celle des routeurs logiciels non virtualisés peut être obtenue en émission et en réception, au prix d'un surcoût en calcul important et d'un partage des ressources non équitable. Nous étudions l'impact des paramètres de l'ordonnanceur de machines virtuelles sur la performance réseau et nous montrons que le module responsable d'acheminer les paquets entre les machines virtuelles et les interfaces physiques constitue le point critique dans les communications réseau.

Mots-clés : Performance réseau, virtualisation, XEN, ordonnancement

Contents

Part I
Short version in english

1 Introduction

Virtualization technology, introduced by IBM in 1973 [1] became very popular with the arrival of systems like Xen [2] and VMware [3]. They are in particular very usefull to enhance isolation, mobility, dynamic reconfiguration and fault tolerance of distributed systems. Virtualization is for example used to run several servers on one physical machine. This method enables organizations and companies to better exploit the physical resources of one machine in terms of CPU and energy: a server uses very often only a small part of the available hardware resources. Moreover, server virtualization allows to increase security due to the isolation factor. By the way, a virtual server can be migrated to another physical machine in case of a breakdown. This guarantees better reliability.
An emerging idea is to implement virtualization technology on the network, inside the routers themselves. Of course, network virtualization already exists in virtual private networks (VPN) which generally use the multi-protocol label switching (MPLS) technology, operating on the link level layer. Another form of virtualization is to segment the physical local area networks into virtual local area networks (VLAN). The virtualization of the "routing machines" of the Internet - the IP routers - opens a new field of possible solutions. With this technology, we could make coexist networks of different types, owners and protocols inside one physical network.
In this report we analyse several performance issues raised by the virtualization mechanisms from the network perspective. Indeed, if virtualization could help in solving main issues of the actual Internet (security, mobility, reliability, configurability), it would nevertheless introduce an overhead due to the additional layers which are placed between the virtual machines and the hardware. Thus, all the network traffic has to cross those virtualization layers to get from the physical interfaces to the virtual space or vice versa, and this impacts the performance. Furthermore, in addition to the routing and bandwidth allocation problems of traditional routers, we have to take into account the problem of local resource sharing by different virtual networks. Considering this sharing of resources like the network interfaces, the processors and the memory, it is a challenge to get a predictable, stable and optimal performance. To have a better insight in these issues, we propose a model of a virtual router we have prototyped with Xen on a software router and we evaluate its properties and its potential. The rest of this report is organized as follows. Section 2 shortly introduces the concept of "virtual infrastructures". Section 3 describes the virtual router model. Section 3.2 discusses some experimental results on network performance considering also the impact of the scheduler configuration on the network performance. Finally, section 5 summarizes some related work and section 6 concludes this report.

2 Virtual Infrastructure

We define an infrastructure as the structured aggregation of computing resources. Thus, a virtual infrastructure represents the aggregation of virtual resources through an organized interconnection network. Figure 10, illustrates the concept of "virtual infrastructures" com-

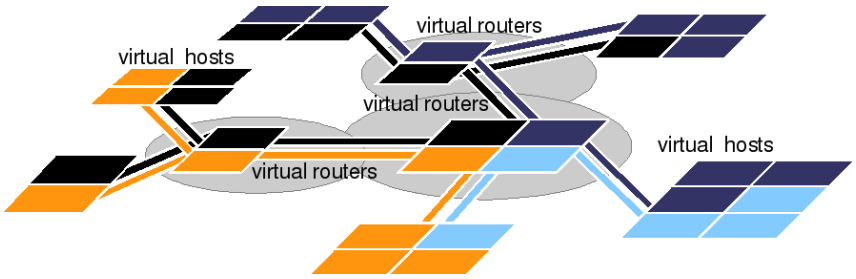


Figure 1: Model of a network with 4 virtual infrastructures interconnected by 3 physical routers containing 8 virtual routers.

posed by the aggregation of virtual machines interconnected via virtual channels as it is proposed by our HIPerNET approach[4]. It shows an example of a network with virtual routers interconnecting 4 virtual infrastructures. The virtual routers can forward independently the traffic of the different virtual infrastructures which share the same physical network. It is known that the virtualization layer enables an efficient separation between the application specifications and physical resources.

3 Virtual router

This section is emphasing the virtual router model which aims at introducing more flexibility, isolation, security and a predictable performance in networks as discussed in the previous section.

3.1 Architecture

The principle of a virtual router is that one physical router is divided into several virtual routers residing inside virtual machines. In a real physical router the retransmission decisions are taken by hardware components while in software routers (i.e. Linux router) they are done by kernel modules. To integrate virtualization, we propose to upload a software router (control and data path) into a virtual machine. The architecture of this virtual router model is represented on figure 2. The principle governing inside the virtual machine is the same than inside a standard software router, except that the virtual machines do not have direct acces to the physical hardware interfaces. The packets have to go through virtual network interfaces to reach the physical interfaces. The packets can be forwarded between

the virtual interfaces and the corresponding physical interfaces thanks to a multiplexing and demultiplexing mechanism implemented in an intermediate layer located between the hardware and the virtual machines.

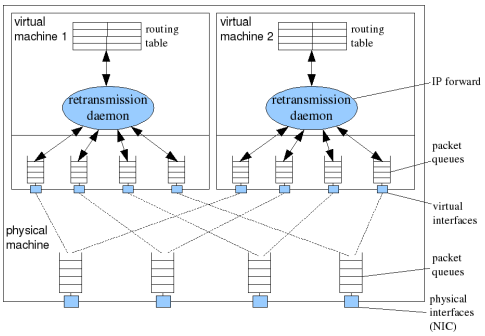


Figure 2: Machine with 2 virtual routers sharing the 4 NICs.

3.2 Performance problem statement

Such a virtual router should support the classical routing and retransmission functions, but also satisfy a certain number of non functional properties like *efficiency*, *fairness* in the resource sharing and *predictability*. We evaluate these properties using the following metrics:
If $N \in \mathbb{N}$ is the number of virtual machines inside the considered physical machine, the metrics are

- R_i the individual throughput of virtual machine $i, i \in \mathbb{N}$
- $R_{aggregate} = \sum_{i=1}^{N} R_i$ the aggregate throughput and $R_{aggregate}/N$ the effective fair rate.
- C_i the CPU cost on virtual machine i
- $C_{aggregate} = \sum_{i=1}^{N} C_i$ the total CPU cost of the system.

To evaluate the overhead of virtualization, we compare the values to different kinds of reference values:

- $R_{classical(T/R))}$ and $R_{classical(F)}$ the throughputs for sendings/receivings and forwardings without virtualization
- $R_{theoretical}$ the maximal theoretical data goodput that can be obtained on the physical interface.

- $C_{classical(T)}$, $C_{classical(R)}$ and $C_{classical(F)}$ the CPU costs of sending, receiving and forwarding on a classical software router.

We define the efficiency in terms of throughput
$E_{throughput} = \frac{R_{aggregate}}{R_{classical}}$.

In order to evaluate the fairness of the inter-virtual machine resource sharing, we calculate the Jain index, defined by the following formula [5]:
$Jain(x) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n \times \sum_{i=1}^{n} x_i^2}$
where n is the number of virtual machines sharing the physical resources and x_i the metric achieved by each virtual machine i, for example the throughput or the CPU percentage.
We evaluate the predictability and scalability analysing the performance according to the number of virtual machines.

3.3 Setup of a virtual router with Xen

We chose to use the Xen technology [2] to implement the virtual router. In Xen, each virtual machine resides in a so called guest domain. Among the guest domains, only a driver domain, in our case domain 0 (dom0), has direct acces to the hardware. All the other domains called domain U (domU) are unprevileged and need to use virtual interfaces.

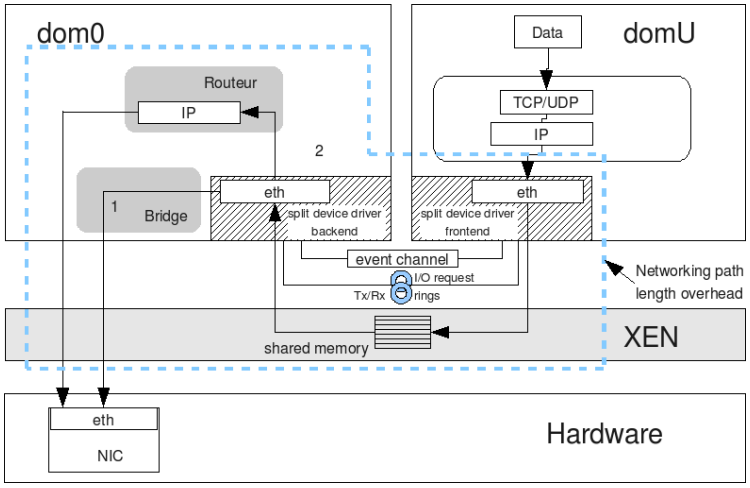


Figure 3: Path of a network packet with Xen, from a domU until the NIC.

The network acces by virtual machines in Xen includes an additional virtualization layer. Each domU has a virtual interface for each physical network interface of the physical machine. This virtual interface can be accessed via a virtual interface *split device driver* composed of two parts, the frontend driver in domU and the backend driver in dom0 [6]. Fig-

ure 12 illustrates the path followed by the network packets emitted on a virtual machine residing inside a domU, then copied to a segment of shared memory by the Xen hypervisor and transmitted to dom0. Inside dom0, packets are bridged (path 1 figure 12) or routed (path 2 figure 12) between the virtual interfaces and the physical ones. The reception of packets on a domU is the same but inversely.
The additional path a packet has to go through due to virtualization is marked by the broken line. A non insignificant processing overhead can be generated due to the additional copy to the shared memory between domU and dom0. Moreover, the multiplexing and demultiplexing of the packets in dom0 can be expensive.
In the following study, we propose to examine the impact of this additional virtualization layer on the networking performance.

4 Experiments and results

4.1 Experimental setup

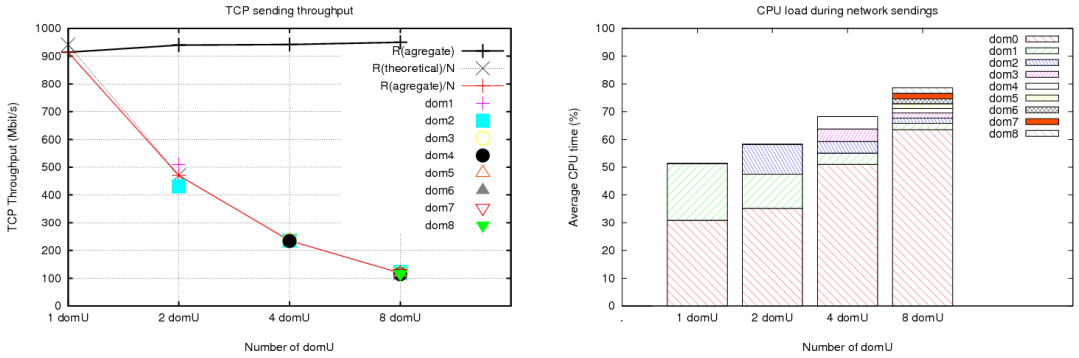
All our experiences are executed on the french national test platform Grid'5000 [7]. We use two types of machines. The tests about sending and receiving on hosts with a single network interface are executed on IBM eServers 325, with 2 CPUs AMD Opteron 246 (2.0 GHz/1MB), one core each one, 2GB of memory and a gigabit ethernet NIC. Tests about forwarding inside virtual routers are executed on IBM eServers 326m, with 2 CPUs AMD Opteron 246 (2.0GHz/1MB), one core each one, 2GB of memory and with 2 gigabit ethernet NICs.
The software configuration used on the IBM eServers 325 is the Xen hypervisor 3.1.0 with the modified 2.6.18-3-xen linux kernel in dom0 and the 2.6.18-3 linux kernel in domU. On the IBM eServers 326m, the software configuration used is the Xen hypervisor 3.0.4 with the modified 2.6.16.33-xen linux kernel in dom0 and 2.6.16.33 kernel in domU.
The machines used in each experience are interconnected via a single switch by gigabit ethernet links (1Gb/s).
The reference values are measured on a Gnu/Linux Debian distribution (Sid) with a 2.6.18 kernel. All the test trafic is generated with the *iperf* tool [8].

4.2 Virtual router performance experiences

4.2.1 Experience 1 : Transmission on virtual hosts

In this first experience, we evaluate the TCP throughput of sendings on 1, 2, 4 and 8 virtual hosts inside one physical machine, and the corresponding CPU overhead. The measured throughputs on each virtual machine, their sum and their average value are represented on figure 4(a). The associated CPU utilization for each guest domain is represented on figure 4(b). The values are averages of the two CPUs.

We notice an aggregate throughput between 914 and 950Mb/s what can be considered as equivalent to the reference throughput $R_{classical(T/R)} = 938Mb/s$ on a linux system without



(a) TCP Throughput

(b) Average processing overhead

Figure 4: TCP Throughput (a) and average processing overhead on both CPUs (b) for sendings on 1, 2, 4 and 8 virtual machines simultaneously.

virtualization. We conclude that the system is efficient in terms of throughput (table 8). For a single domU, both of the CPUs are used at 51%, whereas on a linux system without virtualization, we measured that only $C_{classical(E)} = 32\%$ of both CPUs are in use. With 8 domUs, both of the CPUs are used at 79%.
We notice that even if virtualization introduces a processing overhead, two processors like the ones we use in our experiences are enough to achieve a throughput equivalent to the maximum theoretical throughput on 8 concurrent virtual machines using a 1Gb/s link.

Bandwith is fairly shared between the different domUs. It is the same for the CPU utilization. But dom0 is privileged compared to the domUs. It gets a lot more CPU time. The inter-domU fairness (Jain index) of the throughput and the CPU sharing varies between 0.9932 and 1.0000 (table 8). As these values are close to 1, we conclude that the resource sharing is fair between domUs.

		Number of domUs			
		1	2	4	8
Efficiency	R	0.9744	1.0021	1.0043	1.0138
Fairness (domU)	R		0.9932	1.0000	0.9995
	CPU		0.9955	0.9985	0.9872
Predicta-bility	R	$R_{theoretical}/N$			
	CPU	$\omega_0 = \left(\frac{\ln(N)}{\ln(2)} + 1\right) \times N \times \omega_U$			

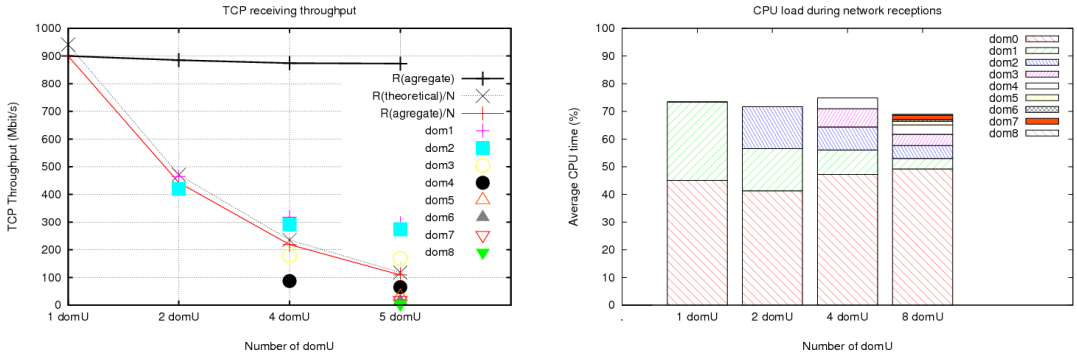
Table 1: Summary of the throughput (R) and CPU results of sendings.

As the total bandwidth is used and fairly shared by the domUs, we conclude that the TCP throughput for sendings is predictable according to the number of virtual machines

and corresponds to the theoretical fair rate $R_{theoretical}/N$, where N is the number of domUs. The CPU utilization seems to follow the formula $\omega_0 = \left(\frac{\ln(N)}{\ln(2)} + 1\right) \times N \times \omega_i$, which applies for our results with 1, 2, 4 and 8 domUs, where ω_0 is the ratio of CPU time used by dom0 according to ω_i, the CPU time used by dom$i$. But this formula should be validated with more experiences.

4.2.2 Experience 2 : Reception on virtual hosts

In this experience, we evaluate the TCP receiving throughput on 1, 2, 4 and 8 concurrent virtual machines and the corresponding processing overhead. Figure 5(a) represents the TCP throughput per domU, the aggregate and the average throughput. Figure 5(b) gives the CPU time distribution among the guest domains. We notice that the aggregate through-



(a) TCP throughput

(b) Average processing overhead

Figure 5: TCP throughput (a) an average processing overhead on both CPUs (b) for receptions on 1, 2, 4 and 8 virtual machines simultaneously.

put decreases slightly according to the number of virtual machines. It attempts 900Mb/s on a single domU and only 873Mb/s on a set of 8 concurrent domUs, what corresponds to 93% of the throughput $R_{classical(T/R)} = 938Mb/s$ on a linux system without virtualization. The efficiency $E_{throughput}$ varies between 0.9303 for 8 domUs and 0.9595 for a single domU. The total CPU cost of the system varies between 69% and 75%, which represents an important overhead compared to a linux system without virtualization, where a network reception takes $C_{classical(R)} = 24\%$. We notice that the efficiency in terms of throughput decreases, but the available CPU time is not entirely consumed.
The sharing of the bandwidth between the domUs is very unfair, especially with a growing number of domUs. The throughput decreases according to the number and so the order of creation of the guest domains. We notice that the distribution of the CPU time among the domUs follows the same pattern. The fairness index decreases until only 0.4832 on 8

concurrent domUs. The unfairness increases according to the number of virtual machines. Table 11 summarizes these results.

		Number of domUs			
		1	2	4	8
Efficiency	R	0.9595	0.9435	0.9320	0.9303
Fairness (domU)	R CPU		0.9974 1.0000	0.8488 0.9298	0.4832 0.6361
Predicta-bility	R CPU	unpredictable			

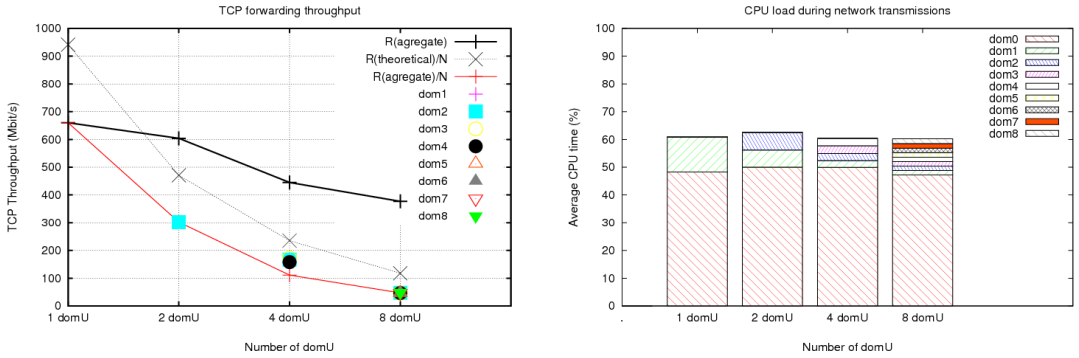
Table 2: Summary of the throughput (R) and CPU results in reception

We conclude that in receptions, the global behaviour of the system is difficult to predict because the resource sharing is unfair. The only thing we notice is that the throughput on one virtual machine depends obviously on the moment of its creation.

4.2.3 Experience 3 : Forwarding on virtual routers

After having evaluated separately the sending and receiving throughput, we evalute the forwarding throughput on virtual machines that forward the traffic between two network interfaces on a machine having two physical interfaces.

The aggregate and individual throughputs on the different virtual machines and their average are represented on figure 6(a). The CPU utilization is represented on figure 6(b). The aggregate TCP throughput of forwarding on virtual routers is much more affected by



(a) TCP throughput

(b) Average processing overhead

Figure 6: TCP throughput (a) and average processing overhead on both CPUs (b) for forwardings on 1, 2, 4 and 8 virtual machines simultaneously.

the virtualization than the sending and receiving throughput of the previous experiences.

It reaches 660Mb/s on a single domU and only 377Mb/s on 8 concurrent domUs, which corresponds to an efficiency between only 70% and 40% of the forwarding throughput on a linux system without virtualization $R_{classical(F)} = 939Mb/s$. The CPU time granted to dom0 is much higher than the CPU utilization of the domUs. Allone, dom0 uses almost 50% of both CPUs, whereas on a classical linux system, the forwarding of a flow needs $C_{classical(T)} = 18\%$ of both CPUs. We observe an important processing overhead, but which does not change even if we increase the number of virtual machines.

The bandwidth and the CPU time are fairly shared between the domUs. The results of the experience are summarized in table 14.

		Number of domUs			
		1	2	4	8
Efficiency	R	0.7029	0.6432	0.4739	0.4015
Fairness (domU)	R		1.0000	0.9975	0.9998
	CPU		1.0000	0.9979	1.9997
Predicta-bility	R CPU	unpredictable			

Table 3: Summary of the throughput (R) and CPU results in forwarding.

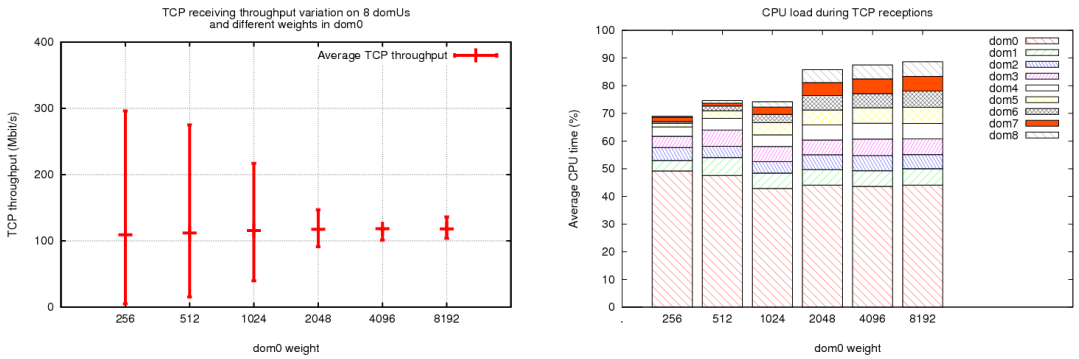
The aggregate throughput on the virtual machines decreases in an irregular manner when we increase the number of virtual machines, so we can not conclude on the predictability and the scalability.

4.2.4 Experience 4 : Configuration of the scheduler

The previous experiences showed that the decrease of the network performance is not due to a lack of processing resources, but rather to the sharing of the processor. So we propose to study the impact of the processor scheduling parameters on the network performance. In the default credit scheduler [9] of Xen, the weight of each guest domain is a configurable parameter. The default value is an amount of 256 credits that each domain consumes while using the processors. A domain which has consumed all its credits enters a low priority state and can be interrupted while using the CPU if an event arrives for another domain being in a state of high priority. Domains that did not yet consume all their credits are in the high priority state. Our experiences showed that dom0 needs much more CPU time than the domUs. Thus we assume that it would rapidly use its credits and enter the low priority state and could be interrupted at each arrival of an event(i. e. packet arrivals) for a domU being in the high priority state. Also, we assume that dom0 has maybe not enough time to treat all the events during its scheduling period, what could be the cause of the unfairness, because the events are treated in the order of the virtual machine creation [10]. Thus we decide to evaluate the network performance with an increased weight of dom0 in order to find out if dom0 is actually the bottleneck of network communications in the credit scheduler's default configuration.

According to the formula we described in section 4, the weight ω_0 of dom0 would depend on the weight of one domU ω_i like $\omega_0 = \left(\frac{\ln(N)}{\ln(2)} + 1\right) \times N \times \omega_i$. The maximum weight of dom0 with 8 domUs would be $\omega_0 = 32 \times \omega_i = 8192$. Therefore we increase the weight of dom0 multiplying it by 2 until 8192 for the receptions. For the forwarding, as we have two physical interfaces and two virtual interfaces in each virtual machine, we consider that the workload of dom0 is 4 times higher, so we increase the dom0 weight until $\omega_0 = 4 \times 8192 = 32768$.

By increasing the weight of dom0, we are able to smooth the TCP throughput variation and the CPU utilization of the different domUs. Figure 7(a) shows an example of those results for simultaneous receptions of TCP flows on 8 domUs inside the same physical machine. The results are similar for 4 domUs. The unfairness becomes very small starting from a weight of 2048 in dom0. Figure 7(b) shows the corresponding CPU utilization. We notice



(a) (b)

Figure 7: TCP throughput variation (a) and CPU time variation (b) on 8 concurrent domUs receiving network flows with different CPU weights in dom0.

a better resource utilization for a weight equal or higher to 2048 and a fair share between the domUs.

For forwardings on virtual routers, by increasing the dom0 weight, we increase the throughput to more than 700Mb/s for 2 and 4 domUs and it reaches 640Mb/s for 8 domUs. It becomes more or less stable starting from a weight equal to 8192. Figure 8 shows these results. The throughput efficiency increases until 0.7066 on 8 domUs compared to 0.4015 in the default scheduler configuration. The CPU utilization does almost not vary by increasing the number of virtual machines. Both, dom0 and domUs benfit from the fact that the percentage of idle CPU decreases.

By increasing the weights of dom0, the model becomes more predictable. In receptions, we can conclude that starting from a weight of 2048 in dom0, the model becomes predictable achieving an aggregate throughput equivalent to the theoretical throughput $R_{theoretical}$. The individual throughput per domU reaches a value close to $R_{theoretical}/N$ for N domUs, with an error of about 20Mb/s. For the forwardings, nevertheless, the model does not become

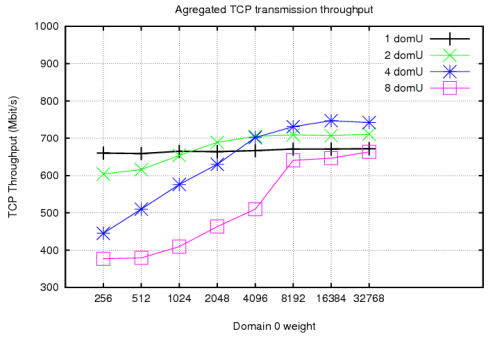


Figure 8: Aggregate TCP throughput in forwardings on domUs with different CPU weights in dom0.

predictable because we can not predict with certitude the aggregate throughput even if it seems to become more or less stable since a certain weight in dom0.

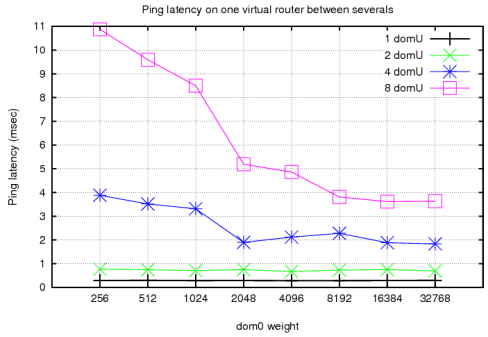


Figure 9: Ping latency on one domU. The other 1, 3 or 7 domUs are forwarding TCP flows.

In order to evaluate the impact of the increasing CPU time ratio attributed to dom0, we measure the end-to-end ping latency between two hosts whose packets are forwarded by virtual routers. Our setup is composed of one routing machine containing respectively 1, 2, 4 or 8 virtual routers forwarding traffic and the corresponding number of remote sender/receiver machine couples. In each test, one virtual router forwards the ping request and response, the other routers forward TCP flows to load the network. The results are presented on figure 9 and show that increasing the dom0 weight improves also the latency.

Finally, the increase of the weight in dom0 leads to improve the efficiency in terms of throughput for the receptions and transmissions on virtual machines. The problem of the unfair resource sharing has also been decreased until a small variation of more or less 20Mb/s. Globally, we observe that the CPU part attributed to dom0 is smaller than it

should be according to the specified weight if this weight is high. We conclude that in this case, dom0 has enough time to forward all the packets waiting between physical and virtual interfaces before a domU gets scheduled. This leads to a better throughput and a better fairness ratio.

5 Related work

Different studies have been done previously to evaluate the network performance provided by Xen. Router virtualization is an emergeant subject. McIlroy and Sventek in [11] propose this concept to separate traffics. A detailed study about the performances of a previous Xen version [12] analyses the network behaviour on guest domains containing a webserver. An evaluation of Xen for network transmissions [13] which aims at studying virtualization for routers shows that the throughput of a transmission on a domU is lower then the one achieved by dom0 on Xen 3.0.4. To avoid problems with context switching overheads, in [13] the author propose all forwarding is handled in the privileged domain (dom0). In other words, domUs are only in charge of control activities of its associated virtual router, while the corresponding forwarding is in dom0. We demonstrate that a change in the scheduling parameters allows better performance also in the domUs.

A comparison of the current Xen schedulers [14] shows that the credit scheduler is interesting for the use of several virtual machines on SMP systems because of its ability to do automatic load balancing. This could be important in networking applications.

The study in [10] confirms our result of the unfair bandwidth distributaion among the domUs in the default configurations of the current Xen versions, with the credit scheduler. The authors propose event channel improvements which enhanced the fairness in the sharing of the bandwidth between the virtual machines. We propose an alternative solution which attributes more CPU time to domain 0. By this method, we also improve fairness.

6 Conclusion

As the virtual intrastructure concept integrates the virtual router paradigm, we then examine the impact of virtualization in sending, receiving and forwarding functions. We showed that network communications in a virtualized environment with Xen need an important amount of processing capacity, especially if the number of virtual machines is important. Our tests included up to 8 uniprivileged virtual machines. We showed that the forwarding throughput decreases in a significant manner. The critical part of the model seems to be the scheduling of the virtual machines and the distribution of the CPU time among them. We showed that dom0 needs much more CPU time than the domUs in network communications, and that privileging dom0 and avoiding too frequent context switchings between the guest domains could perform better throughput. As the performance of each virtual channel is related to the number of virtual machines and the scheduler parameters we will investigate how a self-configuration of the scheduler would offer predictable performance to flows and

virtual infrastructures. We also plan to explore an emerging alternative which is to use virtualization-aware hardware and NICs that actually provide VNICs in hardware.

Deuxième partie
Version détaillée en français

1 Introduction

Les techniques de virtualisation, introduites par IBM dès 1973 [1] sont devenues très populaires avec l'arrivée des systèmes tels que XEN [2] et VMware [3]. Elles sont en particulier très utiles pour accroître l'isolation, la mobilité, la reconfiguration dynamique et la tolérance aux pannes des systèmes distribués. La virtualisation est employée par exemple pour héberger plusieurs serveurs sur une même machine physique. Cette méthode permet aux organisations et aux entreprises de mieux exploiter les ressources physiques d'une machine en termes de CPU et d'énergie : souvent un serveur n'utilise qu'une petite partie de ce qui lui est offert au niveau matériel. De plus, par l'isolation, la virtualisation des serveurs permet d'accroître leur sécurité. Un serveur virtuel peut par ailleurs être migré sur une autre machine physique en cas de panne ce qui assure une meilleure sûreté de fonctionnement.
Une idée émergente est d'employer les techniques de virtualisation au niveau des réseaux, au sein même de leurs routeurs. Certes, la virtualisation dans les réseaux existe déjà dans les réseaux privés virtuels (VPN) et les *Virtual Overlay Networks* (VON) qui utilisent la technologie *Multi-Protocol Label Switching* (MPLS) qui opère au niveau de la couche liaison. Une autre forme de virtualisation est de segmenter les réseaux locaux physiques en réseaux locaux virtuels (VLAN). La virtualisation des "machines de routage" de l'Internet lui-même - les routeurs IP - ouvre un nouveau champ de solutions possibles. Avec cette technique, on peut faire cohabiter des réseaux de types, de propriétaires et de protocoles différents sur un même réseau physique.
Si la virtualisation permettrait de résoudre des questions clés de l'Internet actuel (sécurité, mobilité, fiabilité, configurabilité), elle introduit néanmoins un surcoût dû aux couches supplémentaires s'interposant entre les machines virtuelles et le matériel. Ainsi tout le trafic réseau doit traverser ces couches de virtualisation avant d'arriver dans l'espace utilisateur ou l'espace noyau ou inversément sur l'interface réseau ce qui impacte nécessairement la performance. Par ailleurs, à la problématique du routage et de l'allocation de bande passante dévolue aux routeurs traditionnels s'ajoute le problème du partage des ressources locales par différents réseaux virtuels. Ce partage de ressources comme les interfaces réseau, les processeurs et la mémoire constitue un défi pour obtenir une performance prévisible, stable et optimale.
L'objectif de ce travail est d'explorer cette approche de la virtualisation des réseaux en analysant et évaluant les communications en contexte virtuel. Nous proposons un modèle de routeur ouvert, qui est constitué de plusieurs routeurs virtuels résidant à l'intérieur de machines virtuelles et nous analysons l'impact de la performance d'une implantation d'un tel modèle avec la technologie XEN.
Ce document est organisé de la manière suivante. La première partie présente le modèle. La deuxième partie présente les détails de son implantation sous XEN. La troisième partie

est une étude de performances de cette implantation, suivi par une analyse des résultats et une étude sur l'ordonnancement des machines virtuelles. Une partie suivante propose un changement des paramètres de l'ordonnanceur. La partie 4.2 démontre cette proposition. Enfin la dernière partie développe les travaux relatifs effectués dans le domaine.

2 Routeur virtuel

2.1 Modèle de routeur virtuel

2.1.1 Cas d'application.

Le but de ces travaux est d'étudier la faisabilité d'un routeur virtuel introduisant plus de flexibilité, isolation, sécurité et une performance prévisible dans les réseaux. L'idée est qu'une machine physique servant de routeur est partagée en plusieurs routeurs virtuels résidants dans des machines virtuelles. Nous proposons deux exemples de cas d'application pour ce type de routeurs dans le contexte de l'Internet du futur.

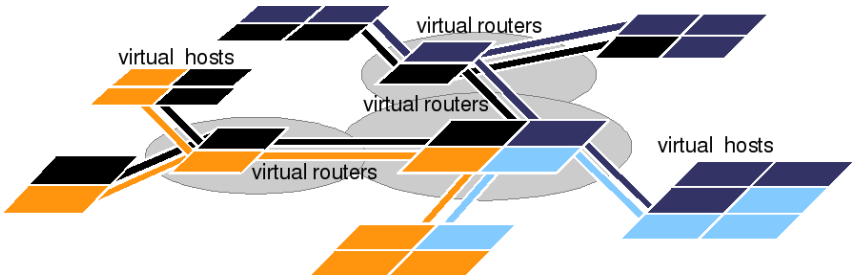


FIG. 10 – Modèle d'un réseau de 4 clusters virtuels interconnectés par 3 routeurs physiques hébergeant 8 routeurs virtuels.

Dans un premier exemple, nous proposons d'utiliser de tels routeurs dans un réseau de clusters virtuels. Cette approche s'inscrit dans le contexte du projet ANR HIPCAL[1] qui étudie un concept de clusters virtuels composés à la demande par interconnexion de machines virtuelles via des canaux virtuels. De tels clusters virtuels peuvent servir pour des applications de calcul nécessitant sécurité, performance et isolation, par exemple des applications distribuées pour l'analyse d'images médicales. La figure 10 représente un exemple de réseau de routeurs virtuels interconnectant 4 clusters virtuels. Des routeurs virtuels peuvent transmettre indépendamment le trafic des différents clusters virtuels qui partagent le même réseau physique. Ils sont ainsi logiquement isolés par la virtualisation et peuvent fournir des services personnalisés aux clusters virtuels au niveau des réservations de bande passante par exemple. L'isolation apporte aussi un niveau de sécurité plus élevé.

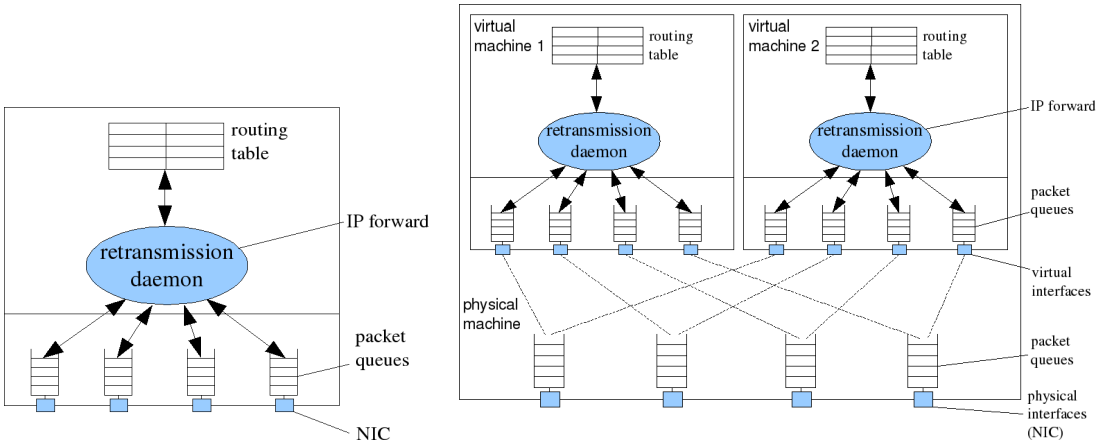
[1] http ://hipcal.lri.fr

Le deuxième exemple d'application est un réseau de routeurs virtuels qui est capable d'adapter le routage et le partage de bande passante aux besoins des applications en termes de qualité de service. La différentiation de trafic peut être mise en œuvre en utilisant un type de routeur virtuel (stratégie de routage, ordonnancement de paquets, etc.) par type de trafic. Par cette méthode, chaque classe de trafic a son propre réseau virtuel. On peut combiner ces réseaux virtuels à l'intérieur du réseau physique de façon à garantir à chaque lien virtuel la qualité de service requise tout en respectant des contraintes. Par exemple la somme des débits garantis sur tous les liens virtuels à l'intérieur d'un même lien physique ne doit pas excéder la bande passante de celui-ci.
D'une manière plus générale, le fait de pouvoir faire cohabiter dynamiquement des réseaux virtuels de caractéristiques différentes grâce aux routeurs virtuels apparaît comme étant une solution prometteuse pour l'Internet du futur visant à découpler l'infrastructure des services [15]. Tous les composants réseau sont identifiés non plus par des adresses IP mais par des identifiants de plus haut niveau et le réseau pourra être vu et contrôlé à travers un "système d'exploitation de réseau" [16].
Afin de construire pas à pas l'Internet du futur, la virtualisation peut servir comme une technique pour développer et tester en environnement réel de nouveaux types de réseaux et de protocoles sans modifier l'infrastructure du réseau, comme l'idée proposée dans *OpenFlow* [17], une technique pour appliquer un traitement différent à des flux différents par des commutateurs hétérogènes ouverts à la configuration personnalisée.

2.1.2 Architecture.

La fonction de base d'un routeur est d'interconnecter plusieurs liens réseau différents et d'acheminer les paquets entre les interfaces réseau. La figure 11(a) montre l'architecture d'un routeur logiciel classique à 4 ports. Son principe de fonctionnement est le suivant : Les paquets reçus sur les interfaces physiques sont mises en file d'attente avant d'être traités par un démon de retransmission qui réside dans l'espace utilisateur. Le démon de retransmission inspecte l'adresse de destination de chaque paquet, consulte la table de routage pour récupérer l'adresse du réseau de destination et achemine le paquet vers l'interface de sortie correspondante. Le principe de fonctionnement d'un routeur réel est identique. Les décisions de retransmission sont par contre prises par des composants matériels et non logiciels comme ici.

Pour intégrer la virtualisation à ce modèle, il est nécessaire de déporter un tel routeur logiciel à l'intérieur d'une machine virtuelle. L'architecture d'un modèle de routeur virtuel est représentée sur la figure 11(b). Le principe de fonctionnement à l'intérieur d'une machine virtuelle est le même que celui du routeur logiciel classique, sauf que les machines virtuelles n'ont pas accès directement aux interfaces physiques. Les paquets sont obligés de passer par des interfaces réseau virtuelles pour atteindre l'interface physique. Un mécanisme de multiplexage et de démultiplexage implémenté dans une couche intermédiaire située entre le matériel et les machines virtuelles permet d'acheminer les paquets entre les interfaces



(a) Routeur de base (b) Routeur virtuel

FIG. 11 – Principe d'un routeur logiciel de base (a) et d'une machine hébergeant deux routeurs virtuels (b).

virtuelles et les interfaces physiques correspondantes.

2.2 Problématique

Un tel routeur virtuel devra supporter les fonctions de routage et de retransmission classiques mais aussi satisfaire un certain nombre de propriétés non fonctionnelles. Ces propriétés sont l'*efficacité*, l'*équité* du partage des ressources inter-machines virtuelles et la *prédictibilité*.

Nous évaluons ces propriétés à l'aide des métriques suivantes :
Soit $N \in \mathbb{N}$ le nombre de machines virtuelles cohabitant dans la machine physique qui constitue notre point de mesure, les métriques sont

- R_i le débit individuel que l'on mesure sur chaque machine virtuelle $i, i \in \mathbb{N}$
- $R_{agrégé} = \sum_{i=1}^{N} R_i$ le débit agrégé mesuré sur l'ensemble des N machines virtuelles.
- $R_{agrégé}/N$ que nous appelons *fair rate* effectif
- C_i le coût CPU par machine virtuelle
- $C_{agrégé} = \sum_{i=1}^{N} C_i$ le coût CPU total du système

Afin d'évaluer le surcoût de la virtualisation, nous comparons les valeurs mesurées à différents types de valeurs de référence :

- $R_{classique(E/R)}$ et $R_{classique(T)}$ qui sont respectivement le débit mesuré en émission ou réception, et en transmission sur un système linux sans virtualisation
- $R_{théorique}$ le *goodput* théorique maximal des données que l'on peut obtenir sur un lien Gigabit Ethernet à 1Gb/s et dont le calcul est détaillé en section 3.1

– $C_{classique(E)}$, $C_{classique(R)}$ et $C_{classique(T)}$ qui sont respectivement le coût CPU d'une émission, réception et transmission sur un système linux classique

Les débits agrégés obtenus de bout en bout sur l'ensemble des machines virtuelles et le coût CPU engendré permettront d'évaluer le taux d'efficacité en termes de débit $E_{débit} = \frac{R_{effectif}}{R_{classique}}$ et en termes de surcoût CPU $E_{CPU} = \frac{C_{classique}}{C_{effectif}}$ où $R_{effectif}$ est le débit agrégé mesuré et $C_{effectif}$ le coût CPU total du système.

Pour évaluer l'équité du partage des ressources inter-machines virtuelles, nous calculons l'index de Jain [5]. La fonction de calcul de l'index de Jain prend en paramètre les mesures individuelles obtenues sur chaque machine virtuelle et retourne une valeur comprise entre 0 et 1 qui indique le taux d'équité. Plus la valeur obtenue est proche de 1, plus le partage est équitable. L'index de Jain est défini par la formule [5] :
$Jain(x) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n \times \sum_{i=1}^{n} x_i^2}$
où n est le nombre d'utilisateurs de la ressource, c'est-à-dire ici le nombre de machines virtuelles recevant, émettant ou transmettant un flux et x_i la valeur de la métrique obtenue par l'utilisateur (la machine virtuelle) i, par exemple son débit ou son pourcentage de CPU. Nous évaluons la prédictibilité et résistance au facteur d'échelle en analysant les performances en fonction du nombre de machines virtuelles.

Cette étude a pour objectif d'inspecter le comportement d'un routeur virtuel en fonction des propriétés annoncées en se basant sur les métriques définies ci-dessus. Nous proposons d'implanter le modèle d'un tel routeur à l'aide de la technologie XEN et nous étudions les questions suivantes :

– Quelle est l'efficacité en termes de débit offerte sur des machines virtuelles comparée celle offerte par un système linux sans virtualisation et quel est le surcoût CPU engendré ?
– Est-ce que le partage des ressources physiques entre les machines virtuelles cohabitant dans une même machine physique est équitable ?
– Comment évolue la performance en fonction du nombre de machines virtuelles concurrentes et est-ce que cette évolution est prédictible afin de permettre un passage à l'échelle ?

2.3 Implantation du routeur virtuel avec XEN

2.3.1 Techniques de virtualisation.

Il existe différentes techniques de virtualisation [18], notamment la virtualisation complète et la paravirtualisation et les systèmes à hyperviseur.
La *virtualisation complète* consiste à émuler tout le matériel de la machine physique afin de le rendre accessible aux machines virtuelles. Celles-ci pouvant donc contenir un système d'exploitation non modifié qui utilise les pilotes habituels pour accéder aux périphériques de manière transparente à travers les périphériques émulés de la couche de virtualisation. Un

désavantage de la virtualisation complète est que l'émulation du matériel est coûteuse et la performance limitée car toutes les instructions au matériel émulé doivent être traduites au vol avant d'être exécutées sur le matériel réel.
La *paravirtualisation* implémente un concept plus léger pour l'accès au matériel. Dans un système paravirtualisé, les machines virtuelles utilisent des interfaces spécifiques "virtuelles" pour accéder au matériel. Ces interfaces communiquent avec les pilotes des périphériques pour transmettre les données aux interfaces physiques. Dans ce type de virtualisation le système d'exploitation hôte a besoin d'être porté pour supporter l'accès aux interfaces virtuelles. La performance offerte par la paravirtualisation peut être obtenue à un coût moindre que dans la virtualisation complète car la méthode d'accès au matériel à travers des interfaces virtuelles est plus légère que l'émulation.
Une technique pour améliorer les performances des systèmes paravirtualisés sont les *hyperviseurs*. Un hyperviseur peut être vu comme une couche qui s'interpose entre le matériel et le système d'exploitation des machines virtuelles. Il gère tous les accès au matériel. Le système d'exploitation fait appel à l'hyperviseur par des *hypercalls*. Du fait que tous les accès matériels passent par l'hyperviseur, c'est celui-ci qui contrôle de manière centralisée l'attribution et la répartition des ressources aux machines virtuelles.

Dans le but de mettre en place un routeur virtuel, nous avons choisi d'utiliser XEN qui est un logiciel libre, gratuit et très répandu dans le milieu académique.

2.3.2 Exemple de technologie de virtualisation : XEN.

XEN [2] est un système de virtualisation à hyperviseur. Il fonctionne selon le principe de la paravirtualisation. Ceci signifie que les machines virtuelles utilisent un pilote virtuel au lieu d'un pilote émulé. Celui-ci agit en intermédiaire entre le système d'exploitation de la machine virtuelle et le pilote réel. Dans XEN, les machines virtuelles résident dans des domaines appelés "invités". Il existe deux types de domaines invités, que l'on appelle domaine 0 (ou dom0) et domaines U (ou domU). Le domaine 0 est unique et privilégié. C'est lui qui gère tous les autres domaines qui sont appelés domaines U où "U" signifie "Unprivileged". Le domaine 0 a un accès direct au matériel et s'occupe par exemple des transferts de données entre les interfaces virtuelles des domaines U et les interfaces physiques des périphériques. Le nombre de machines virtuelles supporté par XEN sur un seul système peut aller jusqu'à 100 [2].
Pour la mise en œuvre d'un routeur virtuel, nous allons étudier plus en détail l'architecture réseau implémentée dans XEN.

2.3.3 Implantation des protocoles réseau dans XEN.

L'accès au réseau par des machines virtuelles dans XEN comporte une couche supplémentaire de virtualisation. Chaque domaine U possède une interface virtuelle pour chaque interface physique de carte réseau de la machine physique. Cette interface virtuelle est accessible via un *split device driver* d'interface virtuelle qui est constitué de deux parts, le *frontend driver* en domaine U et le *backend driver* en domaine 0 [6]. La figure 12 illustre le

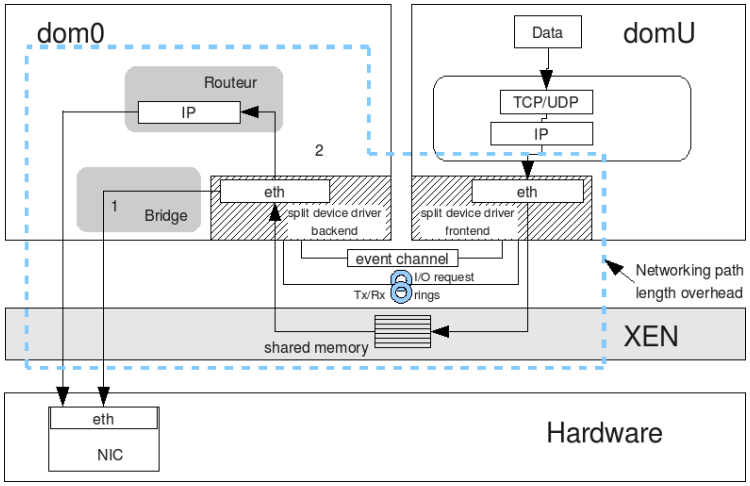


FIG. 12 – Parcours d'un paquet réseau avec XEN, en émission depuis en domaine U jusqu'à la carte réseau.

chemin suivi par les paquets réseau en émission depuis une machine virtuelle résidant dans un domaine U. Les paquets envoyés sur le réseau traversent l'interface virtuelle du domaine U, puis sont copiés dans un segment de mémoire partagée par l'hyperviseur XEN et ensuite transmis dans le domaine 0. Deux buffers circulaires à jetons contiennent les requêtes d'envoi (Tx) et de réception (Rx) de paquets. Un canal d'événements permet de signaler la présence d'un paquet à envoyer ou à recevoir. A l'intérieur du domaine 0, les paquets en provenance des différentes machines virtuelles sont multiplexés et acheminés vers l'interface physique de la carte réseau. L'acheminement se fait par défaut par un pont au niveau de la couche liaison (chemin 1 sur la figure 12). L'alternative est de mettre en place un routeur en domaine 0 dans quel cas les paquets sont remontés dans la pile réseau jusqu'au niveau de la couche réseau pour être acheminés (chemin 2 sur la figure 12).

La réception de paquets sur une machine virtuelle suit le même mécanisme mais dans le sens inverse. Les paquets sont reçus sur l'interface physique et démultiplexés par le domaine 0 qui les achemine ensuite vers l'interface virtuelle du domaine U de destination. L'hyperviseur copie un paquet reçu dans le segment de mémoire partagée et le transmet au domaine U.
Le chemin supplémentaire dû à la virtualisation qu'un paquet doit parcourir pour atteindre la carte réseau depuis un domaine U ou inversément est encadré en pointillé sur la figure 12. Un surcoût de CPU non négligeable peut-être engendré par la copie supplémentaire entre domaine U et domaine 0 dans la mémoire partagée. De plus, le multiplexage et démultiplexage de paquets en domaine 0 peut être coûteux, en particulier dans la version routée, où le traitement du paquet remonte jusqu'à la couche réseau.

Dans l'étude suivante, nous proposons d'étudier l'impact de cette couche supplémentaire sur la performance réseau.

3 Etude expérimentale des performances

3.1 Méthodologie

Avant de détailler les expériences, nous décrivons les caractéristiques du trafic réseau de test, les outils et la plateforme de test.

3.1.1 Protocole et débit.

Nous calculons le *goodput* théorique des données en TCP $R_{théorique}$ sur un lien gigabit ethernet qui représente une valeur de référence pour le débit maximal atteignable.
On suppose que le débit théorique ethernet R_{EthB} est de $10^9 b/s$. La trame MAC est composée d'une préambule et d'un *start frame delimiter* (8 octets), des adresses source et destination (12 octets), du checksum (FCS) (4 octets) et des données. Le tableau 4 montre la composition d'une trame contenant un segment TCP. Les données transmises effectivement dans un segment TCP sont de 1448 octets par trame de taille totale de 1514+4+12+8 octets. Le *goodput* TCP théorique est donc égal à $R_{théorique} = (1448/(1514+4+12+8)) * R_{EthB} = 941.49 Mb/s$. Nous appelons *fair rate* théorique le *goodput* maximal théorique divisé par le

	En-tête + Données						Taille
Trame MAC	24	Données (1514)					1514+4+12+8
Trame Ethernet		14	Données (1500)				1514
Paquet IP			20	Données (1480)			1500
Segment TCP				32	Données (1448)		1480

TAB. 4 – Composition en octets d'une trame contenant un ségment TCP.

nombre de machines virtuelles, soit $R_{théorique}/N$. Cette valeur est équivalente au débit que chaque machine virtuelle devrait achever dans un cas idéal, sans pertes et retransmissions.

3.1.2 Génération de charge.

Pour toutes nos expériences, nous générons du trafic de test à l'aide de l'outil de benchmark *iperf* [8]. Iperf est un programme client/serveur qui permet d'envoyer des flux TCP ou UDP du client vers le serveur et de mesurer le *throughput* du côté client et le *goodput* du côté serveur. En TCP, le trafic est automatiquement injecté jusqu'à la congestion et permet de mesurer le débit maximal que l'on peut obtenir. En UDP, la vitesse d'émission peut être paramétrée.
Pour simuler des calculs et créer une charge CPU artificielle, nous avons développé un outil qui permet d'occuper un processeur avec un calcul arithmétique pendant un pourcentage de temps paramétrable. La charge CPU par les différentes machines virtuelles en XEN est

mesurée à l'aide de l'outil de statistiques *XenMon* [19]. XenMon permet entre autre de mesurer le taux d'occupation par processeur et par domaine invité et le taux de processeur inactif en temps réel sur des intervalles de temps que l'on peut spécifier. Sur un système linux de référence, nous utilisons la commande *sar* pour mesurer le taux d'occupation des processeurs.

3.1.3 Plateforme expérimentale.

Toutes les expériences sont effectuées sur la plateforme nationale de test Grid'5000 [7]. Deux types de machines sont utilisés. Des tests sur des hôtes à une interface réseau en émission et en réception sont exécutés sur des machines de type IBM eServer 325, à deux CPU AMD Opteron 246 (2.0GHz/1MB) à un coeur chacun, 2 GB de mémoire et une carte réseau Gigabit Ethernet (pilote tg3). Des tests sur la performance en transmission à l'intérieur de routeurs virtuels sont effectués sur des machines à deux interfaces réseau Gigabit Ethernet (pilote tg3) de type IBM eServer 326m, à deux CPU AMD Opteron 246 (2.0GHz/1MB) à un cœur chacun et 2 GB de mémoire.
Sur les IBM eServer 325, la configuration logicielle utilisée est l'hyperviseur XEN 3.1.0 avec le noyau linux modifié 2.6.18-3-xen en domaine 0 et le noyau linux 2.6.18-3 en domaine U. Sur les IBM eServer 326m, la configuration logicielle utilisée est l'hyperviseur XEN 3.0.4 avec le noyau linux modifié 2.6.16.33-xen en domaine 0 et le noyau 2.6.16.33 en domaine U. Les machines intervenant dans chaque expérience sont interconnectées par un lien Gigabit Ethernet 1Gb/s et se situent sur un même commutateur.
Les expériences de référence sont menées sur une distribution Gnu/Linux Debian (Sid) avec un noyau 2.6.18.

3.2 Expériences et résultats

Cette étude a pour but d'évaluer les performances réseau dans un contexte de virtualisation, afin d'inspecter les propriétés d'un routeur virtuel décrit précédemment. Nous avons mené plusieurs expériences pour déterminer dans un premier temps les débits TCP entre hôtes virtuels en émission et en réception, le coût CPU associé et l'impact du type de multiplexage de paquets en domaine 0, pont ou routeur. Dans un deuxième temps nous avons évalué le comportement réseau d'un hôte virtuel en émission et en réception sur une machine chargée par du calcul CPU, et enfin, nous avons mesuré le débit TCP en transmission sur un routeur virtuel et le coût CPU associé. Les modèles de génération de trafic sont représentés sur la figure 13.
Le tableau 5 récapitule les différentes expériences, le nombre de machines virtuelles et le nombre de flux générés par expérience et le type de multiplexeur en domaine 0 pour acheminer les paquets.

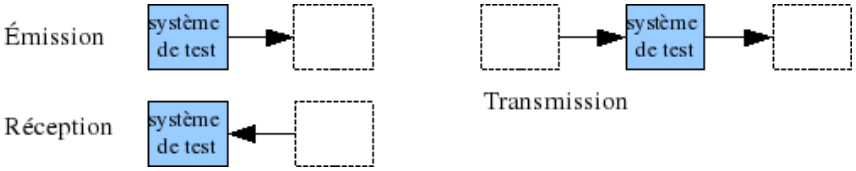


FIG. 13 – Trois types d'expériences, en émission, réception et transmission sur un système virtualisé. Les mesures sont effectuées sur le système de test, les rectangles blancs représentent les générateurs ou puits de trafic.

	Nb de machines virtuelles				Acheminement dom0	
	1	2	4	8	Pont	Routeur
Emission	1xTCP	2xTCP	4xTCP	8xTCP	X	X
Réception	1xTCP	2xTCP	4xTCP	8xTCP	X	X
Emission Charge		1xTCP 1x90%	1xTCP 3x90%	1xTCP 7x90%	X	X
Réception Charge		1xTCP 1x90%	1xTCP 3x90%	1xTCP 7x90%	X	X
Transmission	1xTCP	2xTCP	4xTCP	8xTCP	X	

TAB. 5 – Récapitulatif des expériences.

3.2.1 Expérience 1 : Emission depuis des hôtes virtuels.

But de l'expérience. Cette première expérience vise à évaluer le débit TCP atteignable sur une ou plusieurs machines virtuelles partageant une même machine et un même lien physiques et le coût CPU associé afin de conclure sur l'efficacité du modèle en termes de débit et de surcoût CPU et de la prédictibilité de ces deux facteurs de performance en fonction du nombre de machines virtuelles partageant interfaces réseau et processeurs. Un deuxième but est d'évaluer l'équité de ce partage parmi les machines virtuelles. Enfin, nous comparons les résultats obtenus avec un pont et un routeur en domaine 0 qui achemine les paquets entre l'interface physique et les interfaces virtuelles.

Mise en œuvre. Pour mesurer les performances réseau en émission, nous proposons un système de test qui est composé d'une machine physique que nous appelons machine de test. Elle héberge 1, 2, 4 ou 8 machines virtuelles dont chacune émet un flux réseau. Les flux sont destinés à respectivement 1, 2, 4 ou 8 machines virtuelles distantes. Celles-ci résident dans des machines physiques distinctes afin de limiter l'étude sur le partage des ressources à la machine de test. Un exemple de cette architecture de test avec 4 machines virtuelles est représenté sur la figure 14. Nous envoyons simultanément un flux TCP depuis chaque machine virtuelle de la machine de test vers une machine virtuelle distante. Les mêmes tests sont effectués avec un pont en domaine 0, puis avec un routeur en domaine 0.

Résultats. Les débits mesurés sur chaque machine virtuelle, leur somme et leur moyenne sont représentés sur la figure 15(a). La consommation de CPU associée pour chaque domaine

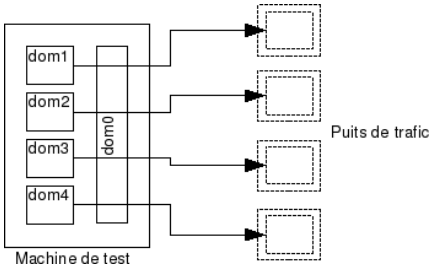
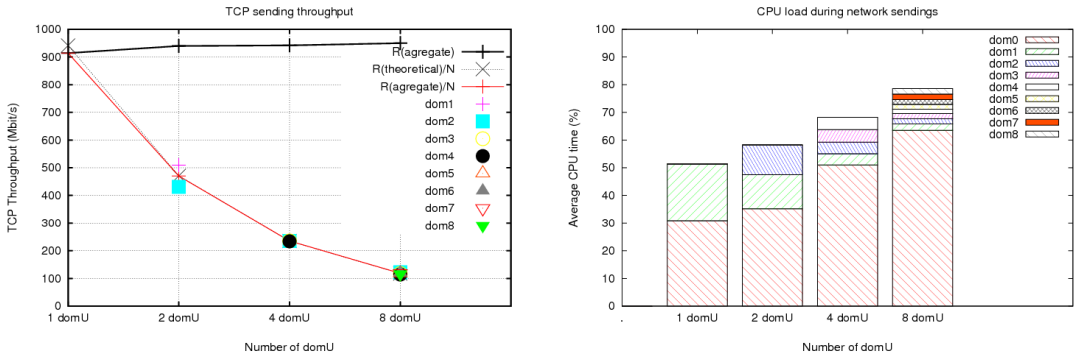


FIG. 14 – Architecture du système de test avec 4 machines virtuelles en émission simlutanée d'un flux.

invité, en moyenne des deux CPU est représentée sur la figure 15(b). Ces résultats ont été obtenus avec un pont en domaine 0. Les résultats obtenus avec un routeur en domaine 0 ne sont pas représentés sur les graphiques car ils sont équivalents. Toutes les valeurs de débits obtenues figurent dans le tableau 6, les mesures de consommation de CPU associées sont représentées dans le tableau 7.



(a) Throughput TCP

(b) Consommation CPU moyenne

FIG. 15 – Throughput TCP (a) et consommation moyenne des deux CPU (b) en émission sur 1, 2, 4 et 8 machines virtuelles simultanément exécutés en utilisant un bridge en domaine 0.

Efficacité : On constate un débit agrégé compris entre 914 et 950Mb/s qui peut être considéré comme équivalent au débit de référence $R_{classique(E/R)} = 938Mb/s$ sur un système linux sans virtualisation. Le système est donc efficace en termes de débit en émission avec un taux d'efficacité $E_{débit}$ compris entre 0.9744 pour une machine virtuelle et 1.0138 pour 8 machines virtuelles, mais ce débit est obtenu à un coût supplémentaire en calcul qui augmente avec le nombre de machines virtuelles. L'efficacité supérieure à 1 pour l'émission sur 8 machines virtuelles s'explique par le fait que l'envoi de plusieurs flux TCP simultané-

		1 domU	2 domU	4 domU	8 domU
Bridge	dom1	914	509	237	123
	dom2		431	235	122
	dom3			236	118
	dom4			234	114
	dom5				118
	dom6				120
	dom7				119
	dom8				116
	cumul	914	940	942	950
	fair rate effectif	914	470	235.5	118.75
	fair rate théorique	941.49	470.75	235.37	117.69
	Jain index		0.9932	1.0000	0.9995
Router	dom1	940	436	237	122
	dom2		507	236	119
	dom3			234	119
	dom4			235	119
	dom5				120
	dom6				116
	dom7				117
	dom8				116
	cumul	940	943	942	948
	fair rate effectif	940	471.5	235.5	118.5
	fair rate théorique	941.49	470.75	235.37	117.69
	Jain index		0.9944	1.0000	0.9997
Linux	938				

TAB. 6 – En émission : Throughput TCP par domaine U et cumulé en Mbit/s sur 1, 2, 4 ou 8 hôtes virtuels simultanément utilisant un bridge ou un router en domaine 0.

ment atteint un débit agrégé plus important que l'envoi d'un seul flux. Pour une machine virtuelle, les deux CPU sont utilisés à 51%, alors que sur un système linux sans virtualisation, seulement $C_{classique(E)} = 32\%$ du temps CPU sont utilisés. Avec 8 machines virtuelles, les deux CPU sont utilisés à 79%. Le taux d'efficacité correspondant E_{CPU} varie entre 0.4133 pour 8 machines virtuelles et 0.6319 pour une machine virtuelle.
Même si la virtualisation entraîne un surcoût en termes de consommation de CPU, deux processeurs tels qu'ils sont utilisés dans nos tests sont suffisants pour achever un débit équivalent au débit théorique maximal avec 8 machines virtuelles concurrentes.

Equité : La bande passante est équitablement répartie parmi les différents domaines U. Il en est de même pour la répartition du temps CPU. Par contre, le domaine 0 est très favorisé par rapport aux domaines U dans le partage du temps CPU. Le taux d'équité (index de Jain) du débit et du partage de CPU varie entre 0.9932 et 1.0000. Les valeurs par nombre de machines virtuelles sont données dans le tableau de synthèse 8. Comme les valeurs sont très proches de 1 on conclut que le partage de la bande passante et des processeurs est très équitable.

Prédictibilité : Puisque la bande passante est totalement consommée par les machines virtuelles et équitablement répartie, on peut conclure que le débit TCP en émission est prédictible en fonction du nombre de machines virtuelles et qu'il correspond environ au *fair*

Bridge		1 domU	2 dom2	4 domU	8 domU
	dom0	30.85425	35.19465	51.0179	63.512400
	dom1	20.55175	12.2725	4.012285	2.34779
	dom2		10.7222	4.207360	1.865295
	dom3			4.510285	1.80796
	dom4			4.432285	1.62541
	dom5				1.751155
	dom6				1.68764
	dom7				2.02616
	dom8				1.950345
	idle	47.50635	37.72925	9.669015	5.60102
	total	98.91235	95.9186	77.84913	84.175175
	Jain index (domU)		0.9955	0.9985	0.9872
Router	dom0	32.8255	35.2537	48.76445	61.1286
	dom1	20.4876	10.9591	4.97375	1.987325
	dom2		12.17855	4.601405	1.774685
	dom3			4.75132	2.31502
	dom4			4.682315	2.423925
	dom5				1.896785
	dom6				1.905905
	dom7				1.887305
	dom8				1.566245
	idle	43.4383	37.5771	9.628725	5.418405
	total	96.7514	95.96845	77.401965	82.3042
	Jain index (domU)		0.9972	0.9992	0.9828
Linux	32.48				

TAB. 7 – En émission d'un flux TCP sur chaque domaine U : Pourcentage de consommation CPU (en moyenne des deux CPU) par le domaine 0 et chacun des domaines U.

rate théorique $R_{théorique}/N$, avec et N le nombre de machines virtuelles.
La consommation de CPU par l'ensemble des domaines invités semble augmenter de façon linéaire quand on double le nombre de machines virtuelles en partant de 51% pour 1 machine virtuelle jusqu'à 79% pour 8 machines virtuelles.

		Nb de domaines U			
		1	2	4	8
Efficacité	Débit	0.9744	1.0021	1.0043	1.0138
	CPU	0.6319	0.5583	0.4764	0.4133
Equité	Débit		0.9932	1.0000	0.9995
(inter-domU)	CPU		0.9955	0.9985	0.9872
Prédictibilité	Débit	oui	oui	oui	oui
	CPU	oui	oui	oui	oui

TAB. 8 – Tableau de synthèse de l'expérience 1.

3.2.2 Expérience 2 : Réception sur des hôtes virtuels.

But de l'expérience. Cette expérience vise à évaluer le débit TCP en réception sur une ou plusieurs machines virtuelles cohabitant une même machine physique et le coût en calcul associé. Comme dans l'expérience précédente, les résultats devront permettre de conclure sur l'efficacité du débit et du coût CPU associé, et sur le partage des ressources et la prédictibilité de la performance en fonction du nombre de machines virtuelles. On évalue également l'impact du type de multiplexage en domaine 0.

Mise en œuvre. Le système de test est le même que précédemment, mais cette fois, les flux sont envoyés simultanément depuis les machines distantes vers les machines virtuelles de la machines de test. L'architecture du modèle est donnée par la figure 16.

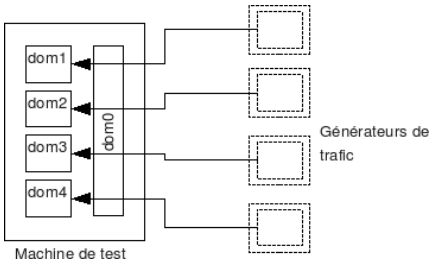
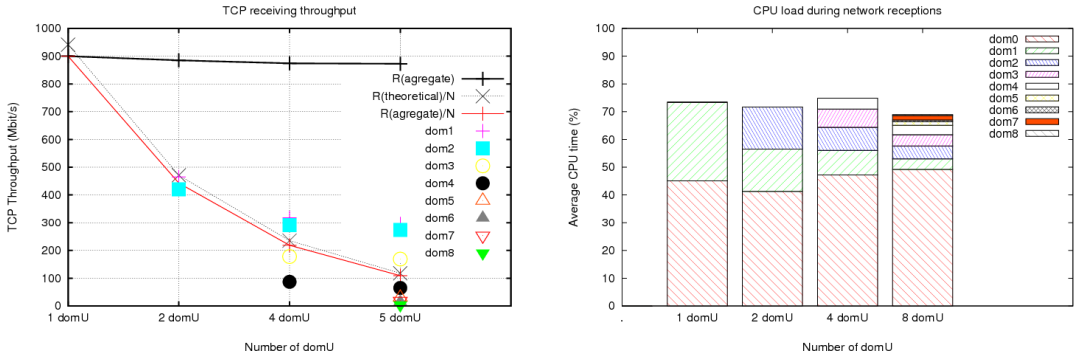


FIG. 16 – Architecture du système de test avec 4 machines virtuelles en réception simlutanée d'un flux.

Résultats. Les resultats sont représentés sur la figure 17. Le graphe (a) représente le débit TCP par machine virtuelle, le débit agrégé et le débit moyen. L'histogramme (b) donne la répartition du temps CPU sur les différents domaines invités. Comme en émission, nous représentons uniquement les résultats obtenus avec un pont en domaine 0 car les résultats obtenus avec un routeur en domaine 0 sont équivalents. Toutes les valeurs mesurées figurent dans les tableaux 9 pour les débits et 10 pour la consommation de CPU associée.

Efficacité : On observe un débit agrégé qui décroît légèrement en fonction du nombre de machines virtuelles. Il est de 900Mb/s pour une seule machine virtuelle et baisse jusqu'à 873Mb/s pour 8 machines virtuelles concurrentes, soit 93% du débit $R_{classique(E/R)} = 938Mb/s$ sur un système linux sans virtualisation. On obtient donc un taux d'efficacité $E_{débit}$ compris entre 0.9303 pour 8 machines virtuelles et 0.9595 pour une machine virtuelle.
La consommation de CPU de l'ensemble des domaines U varie entre 69% et 75%. Le surcoût est très important par rapport au pourcentage consommé sur un système linux sans virtualisation pour la réception d'un flux TCP qui correspond à $C_{classique(R)} = 24\%$. Le taux d'efficacité E_{CPU} varie entre 0.3257 et 0.3538.
On observe qu'en réception, la virtualisation entraîne une baisse en efficacité au niveau du débit, sans pour autant consommer la totalité de la capacité en CPU disponible.



(a) Throughput TCP (b) Consommation CPU moyenne

FIG. 17 – Throughput TCP (a) et consommation moyenne des deux CPU (b) en réception sur 1, 2, 4 et 8 machines virtuelles simultanément exécutés en utilisant un bridge en dom0.

		1 domU	2 domU	4 domU	8 domU
Bridge	dom1	900	465	318	296
	dom2		420	291	274
	dom3			178	169
	dom4			87,2	65,3
	dom5				30,1
	dom6				14,4
	dom7				19
	dom8				4,86
	cumul	900	885	874,2	872,66
	fair rate effectif	900	442.5	218.55	109.08
	fair rate théorique	941.49	470.75	235.37	117.69
	Jain index		0.9974	0.8488	0.4832
Router	dom1	907	444	321	308
	dom2		450	294	229
	dom3			181	170
	dom4			86,7	88,3
	dom5				44,4
	dom6				25,4
	dom7				15,4
	dom8				7,38
	cumul	907	894	882,7	887,88
	fair rate effectif	907	447	220.68	110.99
	fair rate théorique	941.49	470.75	235.37	117.69
	Jain index		1.0000	0.8478	0.5272
Linux	938				

TAB. 9 – En réception sur chaque domaine U : Throughput TCP par domaine U et cumulé en Mbit/s sur 1, 2, 4 ou 8 hôtes virtuels simultanément utilisant un bridge ou un router en domaine 0.

Bridge		1 domU	2 dom2	4 domU	8 domU
	dom0	45.0911	41.27795	47.24035	49.1854
	dom1	28.2269	15.23505	8.8031	3.780645
	dom2		15.1634	8.350072	4.683745
	dom3			6.52665	4.091335
	dom4			3.969005	3.348390
	dom5				1.306877
	dom6				0.608784
	dom7				1.611491
	dom8				0.311964
	idle	24.63955	14.97865	12.50505	8.423375
	total	97.9576	86.6551	87.3942	77.3520
	Jain index (domU)		1.0000	0.9298	0.6361
Router	dom0	44.70895	41.0564	44.86845	49.2445
	dom1	28.8422	14.59605	8.955765	5.97774
	dom2		15.634	9.779755	5.226085
	dom3			6.346255	4.14468
	dom4			4.226535	2.98204
	dom5				1.944225
	dom6				1.111801
	dom7				1.343005
	dom8				0.80264
	idle	23.90665	15.0298	10.3706	7.987505
	total	97.4578	86.31625	84.5474	80.7642
	Jain index (domU)		0.9988	0.9179	0.7161
Linux	24.39				

TAB. 10 – En réception d'un flux TCP sur chaque domaine U : Pourcentage de consommation CPU (en moyenne des deux CPU) par le domaine 0 et chacun des domaines U.

Equité : La bande passante est très inégalement répartie parmi les différents domaines U. On peut observer que le débit décroît en fonction du numéro de domaine. Les domaines sont numérotés dans l'ordre de leur création. Plus un domaine a été créé tôt, plus il semble obtenir une grande part de la bande passante disponible. On peut remarquer que la répartition du temps CPU sur les différents domaines invités suit le même principe.
Le taux d'équité en débit varie entre 0.9974 pour 2 machines virtuelles et seulement 0.4832 pour 8 machines virtuelles. En CPU, le taux d'équité varie entre 1.0000 pour 2 machines virtuelles et 0.6361 pour 8 machines virtuelles. Les valeurs par expérience sont données dans le tableau de synthèse 11. Le partage des ressources n'est pas équitable, en particulier pour 4 et 8 machines virtuelles concurrentes. Le taux d'inéquité augmente avec le nombre de machines virtuelles.

Prédictibilité : Le débit agrégé et la consommation de CPU ne varient que très peu en fonction du nombre de machines virtuelles. Mais le comportement global est difficilement prédictible car le débit de chaque machine virtuelle est très différent. Mais on peut observer que le débit aussi bien que la part de temps CPU d'un domaine U sont liés sa position dans l'ordre de création des machines virtuelles. Le tableau de synthèse 11 récapitule ces résultats.

		Nb de domaines U			
		1	2	4	8
Efficacité	Débit	0.9595	0.9435	0.9320	0.9303
	CPU	0.3327	0.3403	0.3257	0.3538
Equité	Débit		0.9974	0.8488	0.4832
(inter-domU)	CPU		1.0000	0.9298	0.6361
Prédictibilité	Débit	non	non	non	non
	CPU	non	non	non	non

TAB. 11 – Tableau de synthèse de l'expérience 2.

3.2.3 Expérience 3 : Emission et réception réseau sur une machine chargée

But de l'expérience. Dans les expériences précédentes, nous avons vu que le goulot d'étranglement pour les transmissions réseau sur les machines à deux CPU que nous utilisons est la carte réseau dont le débit est limité à 1Gb/s. Dans cette expérience, nous voudrions nous placer dans un contexte où le temps CPU disponible pour les transmissions réseau est limité par une activité de calcul supplémentaire. Nous évaluons alors le débit d'un domaine U sur une machine chargée par du calcul et l'équité dans le partage du temps CPU entre domaines U qui exécutent uniquement des calculs arithmétiques et le domaine U effectuant des communications réseau. Nous évaluons également l'impact du type de multiplexage en domaine 0 dans ces conditions.

Mise en œuvre. Notre système de test comporte une machine de test qui héberge respectivement 2, 4 et 8 hôtes virtuels. Seulement un des hôtes virtuels effectue des communications réseau. Les autres hôtes exécutent chacun un calcul chargeant un CPU pendant 90% du temps qui lui est attribué par l'ordonnanceur. Un exemple de cette architecture de test avec 4 hôtes virtuels est représentée sur la figure 18. Nous mesurons à nouveau le débit

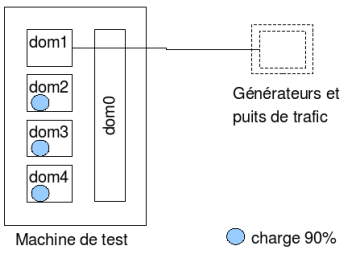
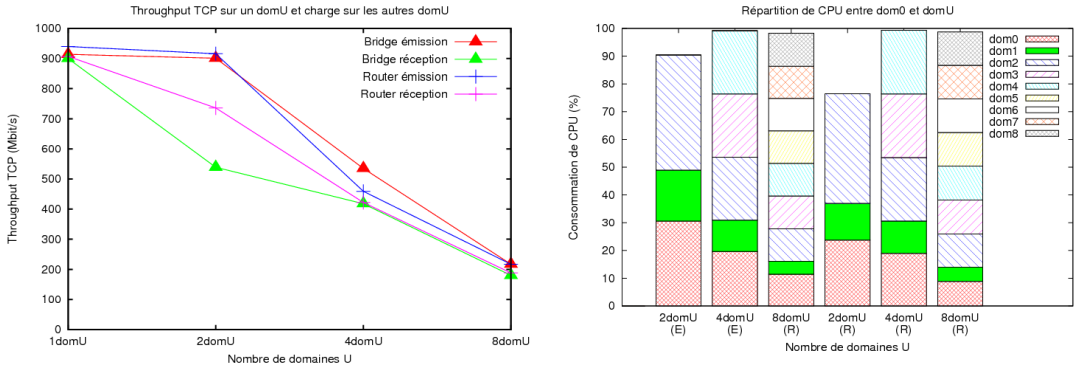


FIG. 18 – Architecture du système de test pour évaluer la performance réseau d'un hôte virtuel cohabitant avec des hôtes virtuels chargeant chacun un CPU à 90%.

TCP en émission et en réception et le coût CPU engendré, en émettant un flux TCP saturant le lien depuis l'hôte virtuel de la machine de test vers une machine virtuelle distante,

puis inversément depuis la machine virtuelle distante vers l'hôte virtuel de la machine de test.



(a) Throughput TCP (b) Consommation CPU moyenne

FIG. 19 – Throughput TCP (a) et consommation moyenne des deux CPU (b) en émission (E) et réception (R) sur 1 machine virtuelle (dom1) cohabitant avec 1, 3 et 7 machines virtuelles exécutant chacune un calul chargeant un CPU pendant 90% du temps qui leur est attribué.

Résultats. Efficacité : La figure 19(a) montre que le débit obtenu en réception et en émission sur le domaine 1 décroît de façon quasi linéaire en fonction du nombre total de machines virtuelles. Le débit sur le domaine 1 décroît jusqu'à 180 Mb/s sous la charge de 7 machines virtuelles exécutant chacune un calcul chargeant un CPU à 90% du temps. Les valeurs associées figurent dans le tableau 12.

	Bridge		Router	
	émission	réception	émission	réception
2 domU	902	539	916	736
4 domU	535	418	459	422
8 domU	217	180	217	188

TAB. 12 – Débit en Mbit/s en émission et réception sur 1 domaine U (dom1) cohabitant avec 1, 3 ou 7 machines virtuelles qui chargent chacune un CPU pendant 90% du temps qui leur est attribué.

En émission le débit ne décroît qu'à partir de 4 machines virtuelles chargeant les processeurs, tandis que le débit en réception est affecté à partir d'une seule machine chargeant le CPU. On note un taux d'efficacité en émission d'un flux $E_{débit(E)}$ compris entre 0.2313 et 0.9616 et un taux d'efficacité en réception d'un flux $E_{débit(R)}$ compris entre 0.1919 et 0.5746. L'efficacité décroît quand le nombre de machines chargeant le processeur augmente. Les résultats sont résumés dans le tableau 14.

La figure 19(b) représente la consommation de CPU (moyenne des deux CPU) par le domaine 0 et les différents domaines U dans la configuration avec un pont en domaine 0.

Comme précédemment, les résultats obtenus avec un routeur en domaine 0 sont équivalents aux résultats obtenus avec un pont en domaine 0. Les valeurs associées sont représentées dans le tableau 13. Dans tous les tests, en émission et en réception, avec 2, 4 et 8 machines

	Emission			Réception		
	2 dom2	4 domU	8 domU	2 dom2	4 domU	8 domU
dom0	30.56855	19.6066	11.4218	23.7223	18.9757	8.8637
dom1	18.35565	11.33535	4.69995	13.22315	11.63825	5.072365
dom2	41.58335	22.62925	11.64525	39.5263	22.8595	12.0246
dom3		22.81575	11.8543		22.9447	12.197
dom4		22.8364	11.7648		22.8886	12.24385
dom5			11.72875			12.1154
dom6			11.67905			12.08675
dom7			11.60035			12.0871
dom8			11.88385			12.0416
idle	8.827095	0.085298	%	20.27945	0.116136	0
total	99.3346	99.3086	98.2781	96.7512	99.422886	98.732365

TAB. 13 – Pourcentage de CPU (moyenne des 2 CPU) attribué au domaine 0 et à chaque domaine U. Ici le domaine 1 émet et reçoit des flux réseaux et les autres domaines U exécutent chacun un calcul chargeant un CPU pendant 90% du temps. L'acheminement des paquets en domaine 0 est effectué par un bridge.

virtuelles, on observe que la somme du temps CPU consommé par le domaine 0 et le domaine 1 est toujours supérieure au temps CPU consommé par un domX avec $X \in [2; N]$. La transmission réseau est donc plus coûteuse en CPU que la charge de calcul. Nous ne calculons pas un taux d'efficacité en surcoût CPU puisqu'en chargeant artificiellement la machine, nous n'autorisons pas de surcoût pour l'activité réseau.

Equité : Le partage des ressources est très équitable. On observe que chaque domaine invité, sauf le domaine 1, reçoit la même part du temps CPU. Mais ceci entraîne une baisse du débit en émission et en réception puisque pour assurer les émissions et réceptions réseau, comme l'ont montré les expériences précédentes, le domaine 0 nécessite plus de temps CPU que la part équitable qu'il reçoit dans le cas de 4 et de 8 machines virtuelles notamment. Le taux d'équité en CPU, E_{CPU} est compris entre 0.8695 et 0.9572. L'inéquité ici détectée est uniquement liée au domaine 1 en émission ou réception réseau qui consomme moins de CPU que les autres domaines.

		Nb de domaines U		
		2	4	8
Efficacité	Débit émission	0.9616	0.5704	0.2313
	Débit réception	0.5746	0.4456	0.1919
Equité	CPU émission	0.8695	0.9451	0.9560
	CPU réception	0.8008	0.9443	0.9572
Prédictibilité	Débit	oui	oui	oui
	CPU	oui	oui	oui

TAB. 14 – Tableau de synthèse de l'expérience 3.

Prédictibilité : On observe que le débit est proportionnel à la part de CPU utilisée par le domaine 0 et le domaine 1 pour l'émission ou la réception du flux. Nous appelons cette part $C_{effectif(charge)}$. Sur un système virtualisé sans charge supplémentaire, comme dans les expériences 1 et 2, nous avons mesurée la consommation de CPU des domaines 0 et 1 en émission ou réception d'un flux sur le domaine 1. Nous appelons cette valeur $C_{virtuel}$. Nous appelons le débit correspondant mesuré dans les expériences 1 et 2 $R_{virtuel}$. Nous observons pour tous les tests de cette expérience que le débit mesuré sur le domaine 1 $R_{effectif(charge)}$ correspond à :
$R_{effectif(charge)} = \frac{C_{effectif(charge)}}{C_{virtuel}} \times R_{virtuel}$
Nous supposons donc que le débit est prévisible en fonction de la part de CPU qui est attribuée aux domaines 0 et 1 qui dépend du nombre de machines virtuelles, et du débit obtenu en context virtualisé sans charge.

Le tableau de synthèse 14 récapitule les résultats obtenus.

3.2.4 Expérience 4 : Transmissions sur un routeur.

But de l'expérience. Après avoir évalué séparément en émission et en réception la performance réseau sur un ensemble d'hôtes virtuels cohabitant une même machine physique, nous inspectons le comportement sur une machine physique avec deux interfaces réseau et des machines virtuelles qui utilisent les deux interfaces pour transmettre des flux entre deux réseaux différents. L'étude d'un tel routeur virtuel simple à deux interfaces permettra de conclure sur les propriétés dans le cas de plusieurs interfaces réseau, des transmissions et réceptions simultanées et un acheminement des paquets entre plusieurs interfaces virtuelles à l'intérieur des domaines U. D'autant plus que nous avons vu que le surcoût se localise en particulier au niveau du domaine 0 et qu'en multipliant les interfaces, la charge du domaine 0 se multiplie proportionnellement.

Mise en œuvre. Nous utilisons une machine de test à deux interfaces physiques et respectivement 2, 4, 8, puis 16 machines physiques distantes hébergeant chacune une machine virtuelle, de façon à ce que chaque machine virtuelle située sur la machine de test transmette le trafic entre deux machines virtuelles distantes. La figure 20 montre un exemple de cette architecture de test avec 4 machines virtuelles.

Résultats Les débits individuels et agrégés des différentes machines virtuelles et leur moyenne sont représentées sur la figure 21(a). Le détail des valeurs figure dans le tableau 15. La consommation de CPU associée est représentée sur la figure 21(b), les valeurs figurent dans le tableau 16.

Efficacité : Le débit TCP agrégé en transmission sur des routeurs virtuels est plus affecté par la virtualisation que le débit en émission ou en réception uniquement. Il s'élève à 660Mb/s pour une machine virtuelle et seulement 377 Mb/s pour 8 machines virtuelles. Pour transmettre un flux sur un domaine U, il n'atteint donc qu'environ 70% du througphut

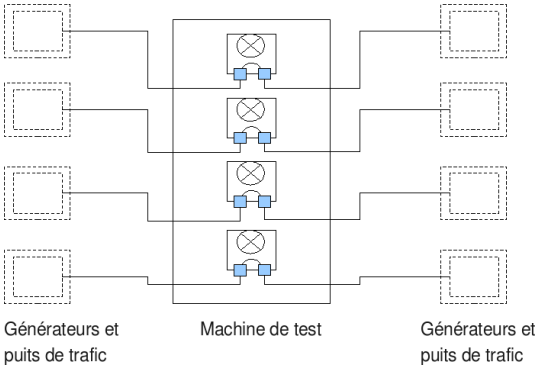
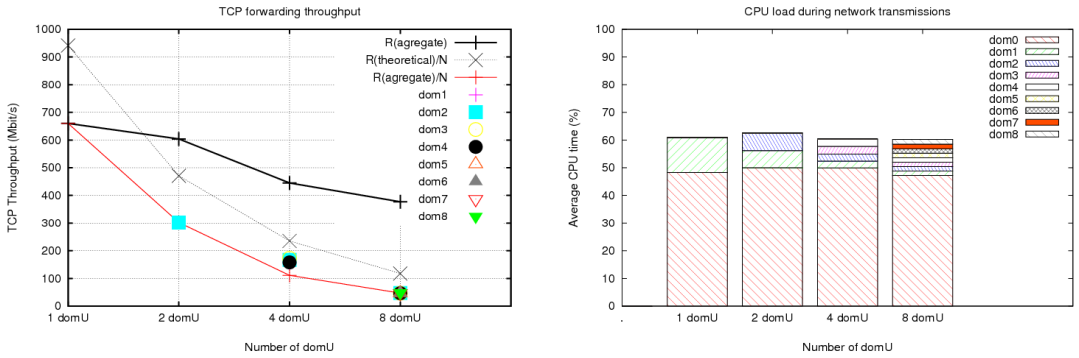


FIG. 20 – Architecture du système de test avec 4 routeurs virtuels en transmission simultanée d'un flux.



(a) Throughput TCP

(b) Consommation CPU moyenne

FIG. 21 – Throughput TCP (a) et consommation moyenne des deux CPU (b) en transmission sur 1, 2, 4 et 8 routeurs virtuels simultanément avec XEN en utilisant un bridge en domaine 0.

TCP en transmission $R_{classique(T)} = 939Mb/s$ sur un système linux sans virtualisation, et seulement 40% dans le cas de 8 machines virtuelles. Le taux d'efficacité $E_{débit}$ varie donc entre 0.4015 pour 8 machines virtuelles et 0.7029 pour une machine virtuelle.
Le temps CPU attribué au domaine 0 est très important comparé à celui attribué aux domaines U. Il atteint presque 50% de la capacité des 2 CPU alors que sur un système linux sans virtualisation, le coût CPU total pour transmettre un flux est d'environ $C_{classique(T)} = 18\%$. Le taux d'efficacité en surcoût CPU, E_{CPU} varie entre 0.2889 et 0.2998. On note donc un surcoût très important mais qui ne varie pas en fonction du nombre de machines virtuelles.

Equité : Le débit et la consommation CPU sont très équitablement répartis parmi les domaines U. Le taux d'équité varie entre 0.9975 et 1.0000 dans tous les tests.

	1 domU	2 domU	4 domU	8 domU
dom1	660	302	102	48.1
dom2		302	113	47.1
dom3			117	46.6
dom4			113	46.5
dom5				46.4
dom6				46.4
dom7				47.5
dom8				48.3
cumul	660	604	445	376.9
fair rate effectif	660	302	111.25	47.1
fair rate théorique	941.49	470.7	235.4	117.7
Jain index		1.000	0.9975	0.9998
Linux	939			

TAB. 15 – Throughput TCP en transmission par routeur virtuel (ou domU) sur 1, 2 ou 4 routeurs virtuels simultanément sur une même machine physique avec un bridge en domaine 0.

	1 domU	2 domU	4 domU	8 domU
dom0	48.26	49.97	49.96	47.23
dom1	12.64	6.22	2.40	1.59
dom2		6.29	2.60	1.60
dom3			2.72	1.59
dom4			2.66	1.61
dom5				1.62
dom6				1.63
dom7				1.65
dom8				1.68
idle	37.16	35.95	38.39	38.82
cumul	98.06	98.43	98.73	99.02
Jain index (dom U)		1.0000	0.9979	0.9997
Linux	18.05			

TAB. 16 – Consommation de CPU de chaque domaine invité en moyenne des 2 CPU lors de la transmission sur 1, 2, 4 ou 8 routeurs virtuels simultanément.

		Nb de domaines U			
		1	2	4	8
Efficacité	Débit	0.7029	0.6432	0.4739	0.4015
	CPU	0.2964	0.2889	0.2991	0.2998
Equité	Débit		1.0000	0.9975	0.9998
(inter-domU)	CPU		1.0000	0.9979	1.9997
Prédictibilité	Débit	non	non	non	non
	CPU	non	non	non	non

TAB. 17 – Tableau de synthèse de l'expérience 4.

Prédictibilité : Le débit agrégé sur l'ensemble des domaines U diminue de façon irrégulière quand le nombre de machines virtuelles augmente. On ne peut pas prédire exactement le

comportement, mais on peut conclure qu'un passage à l'échelle n'est pas possible dans ces conditions. La consommation de CPU totale ne varie que très peu quand on augmente le nombre de machines virtuelles. Le partage des ressources est équitable et correspond au *fair rate* effectif. Les résultats sont résumés dans le tableau de synthèse 17.

3.3 Analyse

Dans les expériences en réception et en transmission de flux, nous avons constaté une baisse de la performance réseau. En réception, le débit agrégé ne diminue que peu quand on augmente le nombre de machines virtuelles, mais une inéquité importante dans le partage des ressources entre domaines invités est à noter. A l'inverse, en transmission sur un routeur à plusieurs machines virtuelles, le débit agrégé diminue fortement pour n'atteindre que 40% du débit en transmission sur un système linux sans virtualisation $R_{classique(T)}$ avec 8 machines virtuelles, mais le partage des ressources est équitable.
Afin de trouver l'origine de l'inéquité dans le partage de la bande passante et des processeurs parmi les différentes machines virtuelles dans le cas de la réception, nous avons effectué des tests afin de vérifier si cette inéquité suit le démarrage des flux et si elle est liée au protocole. La répétition des tests des expériences 1 et 2 en démarrant les flux réseau dans l'ordre inverse de la création des machines virtuelles a donné des résultats équivalents aux résultats initiaux où les flux ont été démarrés dans le même ordre que les machines virtuelles. On conclut donc que la quantité de débit TCP obtenue sur une machine virtuelle dépend de l'ordre de création des machines virtuelles et non des flux. Les mêmes expériences en UDP ont montré que le débit est équitablement réparti parmi les machines virtuelles. L'inéquité rencontrée en réception de flux TCP sur différentes machines virtuelles est donc liée à un surcoût de la virtualisation qui affecte les mécanismes du protocole TCP.

Il est clair que la baisse de performance dûe à l'inéquité du partage des ressources en réception et la baisse du débit agrégé en transmission ne dépendent pas d'un manque de ressources physiques car dans les deux cas, réception et transmission, les deux CPU ne sont pas saturés, même avec 8 machines virtuelles.
D'une manière générale, on remarque que dans tous les cas, émission, réception et transmission, la part de CPU consommée par le domaine 0 est beaucoup plus importante que celle consommée par les domaines U. En émission, on observe que cette part augmente pour le domaine 0 quand le nombre de machines virtuelles augmente, et le débit peut ainsi être maintenu à une valeur équivalente au débit théorique. En revanche, en réception et en transmission, on observe que la part de CPU du domaine 0 n'augmente pas avec le nombre de machines virtuelles. Or plus il y a de machines virtuelles en activité réseau, plus le domaine 0 est chargé pour acheminer les paquets entre les interfaces physiques et les interfaces virtuelles de chaque machine virtuelle. De plus, à chaque acheminement de paquet s'ajoute une copie supplémentaire entre le domaine U et le domaine 0, ce qui demande de nombreux accès à la mémoire et donc un surcoût important en temps. Il est donc possible que la baisse de performance en réception et en transmission soit liée à un manque d'attribution de ressources CPU au domaine 0. Or comme les CPU ne sont pas saturés, on conclut que

le problème provient du fait que leur capacité ne soit pas bien répartie sur les différents domaines invités.
On se propose donc d'étudier de plus près l'ordonnanceur de machines virtuelles qui décide du temps CPU attribué à chaque domaine invité.

3.3.1 Analyse de l'ordonnancement dans XEN.

Dans la version de XEN que nous utilisons, l'ordonnanceur par défaut est le *Credit scheduler* [20, 21]. La particularité du Credit scheduler est qu'il est le premier ordonnanceur dans XEN à effectuer un équilibrage de charge sur les systèmes multi-processeurs (SMP). Cette faculté le rend particulièrement intéressant car il est connu que la virtualisation et notamment les entrées/sorties sont très coûteux en calculs et beaucoup plus efficaces avec plusieurs CPU. Le Credit scheduler implémente des CPU virtuels (VCPU) qui représentent les CPU réels. Ainsi chaque domaine gère les différents CPU de manière transparente en accédant à ses VCPU.
Le fonctionnement du Credit scheduler a été étudié de plus près dans [10]. Il fonctionne sur le principe du partage équitable. Il accorde des parts de temps CPU sous forme de crédits aux différents domaines invités en fonction des poids de ceux-ci. Par défaut, chaque domaine invité a le même poids qui est de 256. Cette valeur correspond à un ratio. Il serait équivalent que chaque domaine possède par exemple 128 crédits. Les domaines invités consomment leurs crédits au cours de l'utilisation de leurs VCPU. Quand ils sont en attente d'utiliser les CPU, ils sont classés dans deux files d'attente de priorités différentes en fonction de leur état qui peut être "OVER" ou "UNDER". Un domaine dans l'état "OVER" a dépassé ses crédits et est classé dans la file la moins prioritaire, alors qu'un domaine dans l'état "UNDER" possède encore des crédits et est donc classé dans la file la plus prioritaire. La file la plus prioritaire est toujours servie en premier. A l'intérieur d'une file, il n'y a pas de notion de priorité quelque soit le nombre de crédits que possède un domaine. Les domaines invités dans une file d'attente sont ordonnancés selon une discipline de type *round robin*.
Une période d'ordonnancement dure 10ms. Au bout de chaque période d'ordonnancement, 100 crédits sont débités au domaine qui est en cours d'utiliser le processeur. Un domaine peut utiliser un processeur pendant au maximum 3 périodes d'ordonnancement successives. Si un domaine n'a pas besoin du processeur pendant la totalité des 30ms, le prochain domaine est ordonnancé. Dans un cas particulier, un domaine X qui est en train d'utiliser le processeur peut être interrompu afin de donner le processeur à un autre domaine Y. Ceci peut se produire lorsqu'un événement à destination du domaine Y arrive. Dans ce cas, l'ordonnanceur réévalue les priorités du domaine X et du domaine Y. Si le domaine X est dans l'état "OVER" et le domaine Y dans l'état "UNDER", l'ordonnanceur décide d'interrompre le domaine X et d'attribuer le processeur au domaine Y.
Ce mécanisme favorise les applications à forte activité d'entrée/sortie comme les communications réseau. Il diminue en fait la latence en accélérant le traitement des événements. Pourtant, l'interruption en fonction des priorités peut aussi engendrer des surcoûts si les priorités sont mal attribuées.

3.3.2 Impact de l'algorithme d'ordonnancement sur les transmissions réseau.

On a vu dans toutes les expériences précédentes que la consommation de CPU par le domaine 0 est beaucoup plus importante que celle des domaines U. D'après le fonctionnement du Credit scheduler expliqué ci-dessus, on peut conclure que le domaine 0 se trouve donc pour la plupart du temps dans l'état "OVER" et est donc le moins prioritaire. En même temps, les domaines U doivent se trouver dans l'état "UNDER" donc le plus prioritaire car ils consomment moins de CPU que la part qui leur est réservée. Ceci signifie que la plupart du temps, lorsqu'un événement à destination d'un domaine U arrive, la situation suivante se produit : Si le domaine 0 est en train d'utiliser le CPU, il est interrompu car il est dans l'état "OVER" et le CPU est attribué au domaine U qui est dans l'état "UNDER" afin que celui-ci puisse traiter l'événement.
Au cours d'émissions réseau depuis un domaine U, les événements à destination des domaines U ne sont pas si fréquents. Ce sont par exemple les événements qui signalent l'arrivée de paquets d'acquittement en TCP. Or en réception sur un domaine U, tous les paquets de données destinés au domaine U sont signalés par événement. Et à chaque arrivée de paquet, si le domaine 0 se trouve dans l'état "OVER", il est interrompu par l'ordonnanceur pour laisser la main au domaine U qui peut donc recevoir le paquet. Ce mécanisme entraînerait donc que le domaine 0 serait interrompu à chaque arrivée de paquet lorsqu'il se trouve dans l'état "OVER". De très fréquents changements de contexte par le CPU peuvent donc conduire à une baisse des performances.

Un autre facteur peut être la gestion des événements décrite par [10]. Un vecteur de bits contient un bit pour chaque domaine invité qui permet de signaler un événement à destination de ce domaine. Les bits sont enchaînés dans l'ordre de création des domaines et le parcours du vecteur pour traiter les événements se fait également dans cet ordre, ce qui peut prioriser les événements des domaines crées plus tôt et expliquer l'inéquité dans le partage des CPU en réception.

Dans le but de contourner l'interruption fréquente du domaine 0 qui pourrait être dûe au fait qu'il se trouve principalement dans un état de priorité plus faible que les domaines U, nous proposons de modifier les paramètres du Credit scheduler.

4 Proposition de configuration de l'ordonnanceur

On voudrait éviter que le domaine 0 soit pour la plupart du temps dans l'état "OVER", le moins prioritaire, alors qu'il est le plus chargé étant donné qu'il doit acheminer les paquets de tous les domaines U. On voudrait que le domaine 0 soit au moins aussi prioritaire que les domaines U à chaque instant. Ainsi les domaines U sont ordonnancés seulement lorsque le domaine 0 a terminé l'acheminement des paquets pour tous les domaines U.
Nous allons proposer un paramétrage de l'ordonnanceur pour influencer sur ces priorités, puis évaluer cette proposition.

4.1 Proposition

Le Credit scheduler permet de spécifier le poids pour chaque domaine invité à la volée. Par défaut celui-ci est de 256. Dans l'expérience en émission de flux réseaux depuis plusieurs machines virtuelles, nous avons obtenu une performance optimale en termes de débit à un certain surcoût en CPU. Nous voudrions donc essayer d'attribuer de façon explicite les poids au domaines invités en fonction de la répartition des poids en émission.
Soit ω_0 le poids du domaine 0 et ω_i le poids d'un domaine U. On observe qu'en émission, la part de CPU attribuée au domaine 0 pour un nombre de machines virtuelles N inférieur ou égal à 8 augmente jusqu'à $4 \times N \times \omega_i$. On décide donc de faire varier ω_0 en émission et réception tel que $\omega_0 \leq 4 \times N \times \omega_i$ (1)
En transmission, le domaine 0 doit acheminer les paquets entre 2 interfaces physiques et 2 interfaces virtuelles dans les domaines U. Le temps CPU qu'il nécessite doit donc s'élever à environ 2x2=4 fois le temps CPU pour l'émission ou la réception sur une seule interface. Nous faisons donc varier le poids en domaine 0 tel que $\omega_0 \leq 4 \times 4 \times N \times \omega_i$ (1)
Avec une valeur maximale de 8 machines virtuelles pour N, nous augmentons alors le poids en domaine 0 par puissances de 2 jusqu'à 8192 en émission et réception sur des hôtes et jusqu'à 32768 en transmission sur les routeurs virtuels.

En augmentant le poids du domaine 0, nous espérons aussi diminuer l'inéquité inter-machines virtuelles rencontrée dans le cas de la réception. Comme ce phénomène semble être lié au parcours du vecteur d'événements toujours dans le même ordre à chaque ordonnancement du domaine 0, une prolongation de la période d'ordonnancement du domaine 0 pourrait permettre de parcourir tout le vecteur d'événements et de traiter les événements à destination de tous les domaines.

4.2 Evaluation

Dans cette partie, nous décrivons la série de tests effectués et leurs résultats pour valider notre proposition. Nous étudions si l'augmentation du poids du domaine 0 par rapport aux domaines U peut améliorer la performance réseau. Toutes les expériences sont effectuées avec un pont pour acheminer les paquets en domaine 0.

4.2.1 Expérience 5 : Impact du paramétrage de l'ordonnanceur sur le débit.

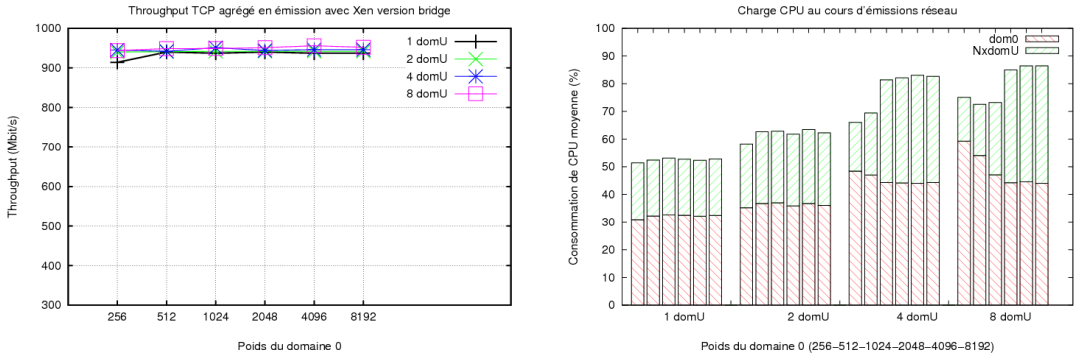
But de l'expérience. L'objectif de cette expérience est de réévaluer les contraintes d'efficacité, d'équité et de prédictibilité en fonction du paramétrage de l'ordonnanceur. Puisque nous avons observé que le domaine 0 nécessite beaucoup de temps CPU pour acheminer les paquets, nous voudrions savoir si l'attribution d'une part de CPU plus importante correspondant aux formules (1) et (2) au domaine 0 peut améliorer les performances réseau.

Mise en œuvre. Nous répétons les mêmes tests que précédemment (expériences 1, 2 et 4), mais en augmentant le poids du domaine 0 à partir de sa valeur par défaut (256) par puissances de 2, jusqu'à 8192 dans les émissions et réceptions et jusqu'à 32768 dans les

expériences en transmission car nous supposons que le domaine 0 nécessite plus de ressources dans le cas de transmissions entre plusieurs interfaces réseau.

Résultats : Efficacité. Dans les 3 scénarios, émission, réception et transmisson, on obtient un débit TCP agrégé qui est croissant quand on augmente le poids du domaine 0. L'augmentation du débit est particulièrement importante pour la transmission.

Emission : En émission, le débit agrégé mesuré atteint déjà quasiment la valeur de référence avec un poids de 256 pour chaque domaine invité. On voit sur la figure 22(a) qu'en augmentant le poids du domaine 0, le débit agrégé reste stable quelque soit le nombre de machines virtuelles. On observe que le temps CPU accordé au domaine 0 (figure 22(b)) diminue au profit des domaines U qui reçoivent plus que deux fois plus de temps processeur. Le détail de toutes les valeurs mesurées en termes de débit et de consommation CPU associée



(a) (b)

FIG. 22 – Throughput TCP agrégé en émission sur 1, 2, 4 et 8 machines virtuelles simultanément en fonction du poids du domaine 0, avec XEN en utilisant un bridge en domaine 0 et un poids de 256 à chaque domaine U.

figure dans les tableaux 18 à 25.

Poids dom0	256	512	1024	2048	4096	8192
dom1	914	940	937	940	937	937
TCP goodput théorique	941.49					

TAB. 18 – Throughput TCP en émission en Mbit/s sur une seule machine virtuelle par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Réception : En réception, avec la configuration par défaut de l'ordonnanceur, le débit diminue légèrement quand le nombre de machines virtuelles augmente (jusqu'à 872 Mb/s pour 8 machines virtuelles). En augmentant le poids du domaine 0 (figure 23(a)), on arrive

Poids dom0	256	512	1024	2048	4096	8192
dom0	30.85	32.19	32.57	32.44	32.12	32.37
dom1	20.55	20.23	20.57	20.27	20.22	20.39
idle	47.51	44.46	45.29	44.16	44.54	44.51
Cumul	98.91	96.88	98.43	96.87	96.88	97.27

TAB. 19 – Consommation de CPU en émission TCP sur 1 domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom1	509	471	419	466	469	471
dom2	431	470	523	476	473	471
cumul	940	941	942	942	942	942
fair rate effectif	470	470.5	471	471	471	471
fair rate théorique	470.75					
Jain index	0.9932	1.0000	0.9880	0.9999	1.0000	1.0000

TAB. 20 – Throughput TCP en émission en Mbit/s sur deux machines virtuelles simultanément par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom0	35.19	36.73	36.95	35.82	36.74	36.04
dom1	12.27	13.10	11.96	12.93	13.38	13.16
dom2	10.72	12.83	13.91	13.02	13.33	13.05
idle	37.43	34.81	35.87	34.12	34.79	34.01
Cumul	95.92	97.47	98.69	95.89	97.57	96.26
Jain index	0.9955	0.9999	0.9944	1.0000	1.0000	1.0000

TAB. 21 – Consommation de CPU en émission TCP sur 2 domaines U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom1	243	238	241	237	236	238
dom2	234	235	235	235	235	236
dom3	234	234	236	236	239	235
dom4	234	235	239	236	236	237
cumul	945	942	951	944	946	946
fair rate effectif	236.3	235.5	237.8	236	236.5	236.5
fair rate théorique	235.37					
Jain index	0.9997	1.0000	0.9999	1.0000	1.0000	1.0000

TAB. 22 – Throughput TCP en émission en Mbit/s par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

à augmenter et à stabiliser le débit TCP à une valeur proche de la valeur de référence $R_{classique(R)} = 938Mb/s$. Le taux d'efficacité $E_{débit}$ augmente à 1.0021 à partir d'un poids de 4096 avec 8 machines virtuelles. Pour la consommation de CPU, les résultats sont similaires à ceux en émission (figure 23(a)). On note que le pourcentage de CPU inactif diminue et que les ressources sont donc mieux utilisées quand on augmente le poids du domaine 0. Le

Poids dom0	256	512	1024	2048	4096	8192
dom0	48.44	47.01	44.26	44.11	44.01	44.30
dom1	4.49	5.71	8.94	9.21	9.36	9.27
dom2	4.41	5.73	9.40	9.46	9.72	9.57
dom3	4.36	5.44	9.37	9.63	10.01	9.70
dom4	4.32	5.55	9.42	9.71	9.94	9.81
idle	10.13	11.47	12.94	13.48	13.69	13.59
Cumul	76.15	80.91	94.33	95.60	96.05	96.24
Jain index	0.9998	0.9995	0.9995	0.9996	0.9993	0.9996

TAB. 23 – Consommation de CPU en émission TCP sur 4 domaines U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom1	133	130	123	114	115	114
dom2	129	127	120	113	114	112
dom3	128	126	121	113	115	112
dom4	129	124	121	112	113	111
dom5	127	126	121	112	114	112
dom6	126	126	121	113	113	111
dom7	45.8	62.8	99.0	158	157	166
dom8	126	127	123	116	115	114
cumul	943.8	948.8	949	951	956	952
fair rate effectif	117.98	118.6	118.63	188.88	119.5	119
fair rate théorique	117.69	117.69	117.69	117.69	117.69	117.69
Jain index	0.9490	0.9692	0.9960	0.9847	0.9861	0.9781

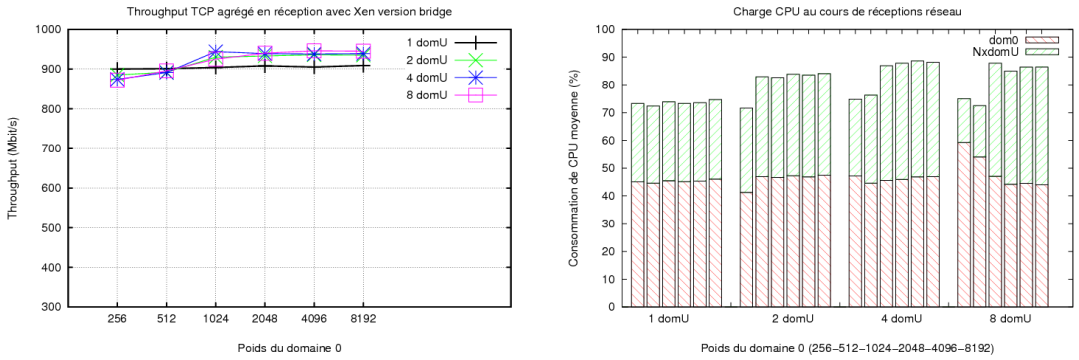
TAB. 24 – Throughput TCP en émission en Mbit/s par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom0	59.24	54.05	47.05	44.23	44.52	43.98
dom1	2.10	2.78	3.35	4.97	5.06	5.18
dom2	2.16	2.53	3.50	4.91	5.07	5.09
dom3	1.81	2.51	3.28	4.95	5.19	5.12
dom4	1.88	2.19	3.28	4.93	5.09	5.15
dom5	2.66	2.31	3.15	5.00	5.14	5.17
dom6	1.85	2.41	3.22	5.05	5.16	5.21
dom7	1.23	1.56	2.97	5.79	5.91	6.20
dom8	2.16	2.19	3.43	5.18	5.26	5.35
idle	6.69	6.88	7.71	7.19	6.92	7.06
Cumul	81.78	79.41	80.94	92.20	93.32	93.51
Jain index	0.9643	0.9791	0.9978	0.9971	0.9975	0.9958

TAB. 25 – Consommation de CPU en émission TCP sur 8 domaines U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

taux d'efficacité E_{CPU} varie peu pour 1 et 2 machines virtuelles mais diminue jusqu'à 0.2751 pour 4 et 8 machines virtuelles avec un poids de 8192 en domaine 0, le surcoût augmente avec le débit.

Le détail des valeurs en termes de débit en réception de de consommation CPU associées aux



(a) (b)

FIG. 23 – Throughput TCP agrégé en réception sur 1, 2, 4 et 8 machines virtuelles simultanément en fonction du poids du domaine 0, avec XEN en utilisant un bridge en domaine 0 et un poids de 256 à chaque domaine U.

différentes expériences (avec 1, 2, 4 puis 8 machines virtuelles) est donné dans les tableaux 26 à 33. On peut observer l'évolution de l'inéquité avec l'augmentation du poids en domaine 0.

Poids dom0	256	512	1024	2048	4096	8192
dom1	900	901	904	908	905	909
TCP goodput théorique	941.49					

TAB. 26 – Throughput TCP en réception en Mbit/s sur une seule machine virtuelle par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom0	45.09	44.64	45.51	45.14	45.36	46.08
dom1	28.23	27.75	28.43	28.18	28.24	28.64
idle	24.64	23.93	24.07	23.99	24.02	23.37
Cumul	97.96	96.32	98.01	97.31	97.62	98.09

TAB. 27 – Consommation de CPU en réception TCP sur 1 domaine U avec le credit-based scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Transmission : En transmission sur des routeurs virtuels, le débit TCP agrégé est très dégradé dans la configuration par défaut de l'ordonnanceur, jusqu'à moins de 400Mb/s pour 8 machines virtuelles. L'augmentation du poids pour le domaine 0 (figure 24(a)) permet d'augmenter ce débit et de le stabiliser à partir d'un poids de 8192 au domaine 0 à des valeurs supérieures à 700Mb/s pour 2 et 4 machines virtuelles et supérieur à 640Mb/s pour

Poids dom0	256	512	1024	2048	4096	8192
dom1	465	454	438	447	470	465
dom2	420	477	492	486	466	470
cumul	885	892	930	933	936	935
fair rate effectif	442.5	446	465	466.5	468	467.5
fair rate théorique	470.75					
Jain index	0.9974	0.9994	0.9966	0.9983	1.0000	1.0000

TAB. 28 – Throughput TCP en réception en Mbit/s sur deux machines virtuelles simultanément par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom0	41.28	46.95	46.71	47.30	46.90	47.48
dom1	15.24	17.40	16.93	17.56	18.35	18.08
dom2	15.16	18.54	18.95	18.96	18.30	18.47
idle	14.98	12.57	12.81	12.62	12.88	13.08
Cumul	86.66	95.46	95.40	96.44	96.43	97.11
Jain index	1.0000	0.9990	0.9968	0.9985	1.0000	0.9999

TAB. 29 – Consommation de CPU en réception TCP sur 2 domaines U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom1	318	295	261	247	242	241
dom2	291	283	249	241	233	239
dom3	178	194	232	239	248	225
dom4	78.2	119	202	212	215	234
cumul	874.2	891	944	939	938	939
fair rate effectif	218.55	222.75	236	234.8	234.5	234.8
fair rate théorique	235.37					
Jain index	0.8488	0.9066	0.9913	0.9967	0.9972	0.9993

TAB. 30 – Throughput TCP en réception en Mbit/s par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom0	47.24	44.62	45.58	45.93	46.89	46.92
dom1	8.80	9.70	10.72	10.45	10.54	10.41
dom2	8.35	9.48	10.97	10.97	10.25	10.43
dom3	6.53	7.23	10.37	10.64	11.08	9.98
dom4	3.97	5.36	9.25	9.84	9.87	10.41
idle	12.51	11.62	8.30	8.08	7.71	8.25
cumul	87.40	88.01	95.19	95.91	96.34	96.40
Jain index	0.9298	0.9523	0.9960	0.9985	0.9982	0.9997

TAB. 31 – Consommation de CPU en réception TCP sur 4 domaines U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

8 machines virtuelles. Un débit maximal de 663Mb/s pour 8 machines virtuelles est atteint

Poids dom0	256	512	1024	2048	4096	8192
dom1	296	275	217	147	139	136
dom2	274	210	169	119	121	113
dom3	169	179	164	131	130	124
dom4	65.3	99.0	121	125	123	120
dom5	30.1	58.9	91.2	118	119	124
dom6	14.4	36.5	70.3	110	106	117
dom7	19	21.9	52.7	98.9	107	107
dom8	4.86	15.3	39.7	91.2	101	104
cumul	872.66	895.6	924.9	940.1	946	945
fair rate effectif	109.08	111.95	115.61	117.51	118.25	118.13
fair rate théorique		117.69	117.69	117.69	117.69	117.69
Jain index	0.4832	0.6001	0.7933	0.9804	0.9896	0.9934

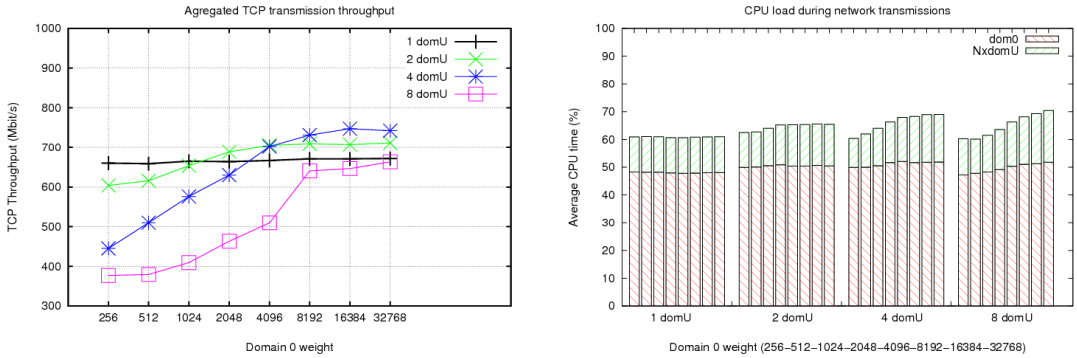
TAB. 32 – Throughput TCP en réception en Mbit/s par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	256	512	1024	2048	4096	8192
dom0	49.19	47.56	42.88	44.06	43.62	44.09
dom1	3.78	6.52	5.55	5.63	5.65	5.94
dom2	4.68	4.05	4.10	5.34	5.46	5.11
dom3	4.09	5.84	5.49	5.32	5.96	5.64
dom4	3.35	4.21	4.20	5.46	5.75	5.59
dom5	1.31	2.79	4.38	5.40	5.59	5.80
dom6	0.61	1.72	3.03	5.24	5.11	5.86
dom7	1.61	0.95	2.66	4.67	5.33	5.36
dom8	0.31	0.96	1.93	4.62	5.00	5.26
idle	8.42	5.40	5.76	4.08	3.61	3.54
cumul	77.35	80.00	79.98	89.82	91.08	92.19
Jain index	0.7068	0.7412	0.9124	0.9957	0.9970	0.9975

TAB. 33 – Consommation de CPU lors de réceptions TCP par domaine U avec le Credit scheduler et différents poids pour le domaine 0 et un poids de 256 pour chaque domaine U.

pour le poids de 32768. Le taux d'efficacité augmente jusqu'à 0.7066 sur 8 machines virtuelles, comparé à 0.4015 avant la modification de poids. Les valeurs de débit mesurées figurent dans le tableau 34.
La consommation de CPU ne varie pas beaucoup (figure 24(b)). Le pourcentage de CPU inactif diminue au profit à la fois du domaine 0 et des domaines U.

Résultats : Equité. L'expérience 2 montre que la répartition des ressources au cours de réceptions TCP sur plusieurs machines virtuelles est très inéquitable dans la configuration par défaut de l'ordonnanceur. L'augmentation du paramètre de poids pour le domaine 0 a pour conséquence un lissage dans la répartition des débits TCP en réception et de la consommation de CPU des machines virtuelles concurrentes. La figure 25 montre un exemple de ces résultats pour la réception de flux TCP simultanée sur 8 machines virtuelles cohabitant dans la même machine physique. Les résultats sont semblables pour 4 machines virtuelles.
Le taux d'inéquité devient minimal à partir d'un poids de domaine 0 égal à 4096. Le *fair*



(a) (b)

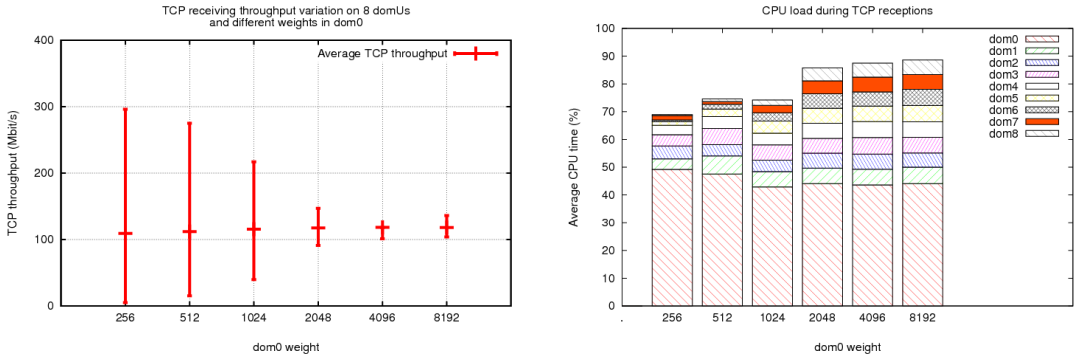
FIG. 24 – Throughput TCP agrégé en transmission sur 1, 2, 4 et 8 machines virtuelles simultanément en fonction du poids du domaine 0, avec XEN en utilisant un bridge en domaine 0 et un poids de 256 pour chaque domaine U.

Poids dom0	1 domU	2 domU	4 domU	8 domU
256	660	604	445	376.9
512	659	616	510	379.4
1024	665	654	576	409.4
2048	664	689	630	463.3
4096	667	705	702	509.6
8192	671	709	731	640.6
16384	671	707	747	646.5
32768	672	711	742	663.5
Linux	939			

TAB. 34 – Evolution du throughput TCP en transmission sur 2 et 4 routeurs virtuels et avec différents poids attribués au domaine 0 et un poids de 256 attribué à chaque domaine U.

rate effectif et les débits individuels s'y rapprochent. On constate une meilleure utilisation des resources CPU plus le poids du domaine 0 est important.

Résultats : Prédictibilité. En général, plus on donne du poids au domaine 0, plus le modèle devient prédictible. En émission, le débit ne varie pas par rapport aux résultats initiaux et le partage est équitable. On peut donc prédire le débit par machine virtuelle. En réception, on peut conclure qu'à partir d'un poids de 2048 pour le domaine 0, le modèle devient prédictible de façon à achever un débit équivalent au débit théorique $R_{théorique}$. Le débit individuel atteint une valeur plus proche de $R_{théorique}/N$ pour N domaines U, à une marge d'erreur d'environ 20Mb/s. En transmission, le débit se stabilise entre environ 640 et 740Mb/s pour 1 à 8 machines virtuelles. Il en est de même pour le surcoût en CPU. Les nouveaux résultats en réception sont représentés dans le tableau 35 et en transmission dans le tableau 36.



(a) Variation throughput TCP

(b) Consommation CPU moyenne

FIG. 25 – Impact des paramètres du Credit scheduler : Variation du débit TCP (a) et consommation CPU (b) en moyenne des 2 CPU sur 8 machines virtuelles recevant des flux TCP simultanément, en faisant varier le poids du domaine 0, le poids de chaque domaine U est de 256.

		Poids du domaine 0					
		256	512	1024	2048	4096	8192
Efficacité	Débit	0.9303	0.9548	0.9860	1.0021	1.0085	1.0075
	CPU	0.3538	0.3269	0.3286	0.2845	0.2788	0.2751
Equité	Débit	0.4832	0.6001	0.7933	0.9804	0.9896	0.9934
(inter-domU)	CPU	0.7068	0.7412	0.9124	0.9957	0.9970	0.9975
Prédictibilité	Débit	non	non	non	oui	oui	oui
	CPU	non	non	non	oui	oui	oui

TAB. 35 – Tableau de synthèse de l'expérience 5 en réception sur 8 machines virtuelles.

		Poids du domaine 0							
		256	512	1024	2048	4096	8192	16384	32768
Efficacité	Débit	0.4014	0.4040	0.4360	0.4934	0.5427	0.6822	0.6884	0.7066
	CPU	0.2998	0.3002	0.2935	0.2838	0.2723	0.2649	0.2605	0.2562
Prédictibilité	Débit	non	non	non	non	non	oui	oui	oui
	CPU	non	non	non	non	non	oui	oui	oui

TAB. 36 – Tableau de synthèse de l'expérience 5 en transmission sur 8 machines virtuelles.

4.2.2 Expérience 6 : Impact du paramétrage de l'ordonnanceur sur la latence.

But de l'expérience. L'objectif de cette expérience est d'évaluer l'impact de la modification des paramètres de poids de l'ordonnanceur sur la latence. En augmentant le poids du domaine 0, on prolonge sa période d'ordonnancement. On voudrait étudier si cette prolongation se reflète aussi dans la latence lors d'émissions ou réceptions sur plusieurs machines virtuelles simultanément.

Mise en œuvre. Pour mesurer la latence, nous utilisons la commande `ping`. Afin d'étudier le pire cas, nous chargeons le plus possible la machine de test avec des émissions/réceptions réseau. En émission, nous envoyons des requêtes de ping depuis une des machines virtuelles de la machine de test et des flux TCP avec iperf depuis les autres machines virtuelles. Les mêmes tests sont répétées en réception, en envoyant les flux dans le sens invers (d'hôtes virtuels distants vers la machine de test), puis en transmission sur des routeurs virtuels. Les

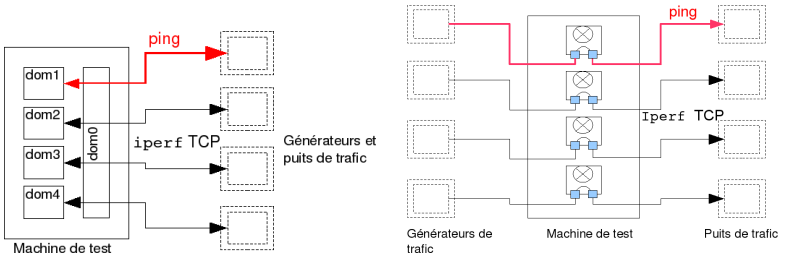
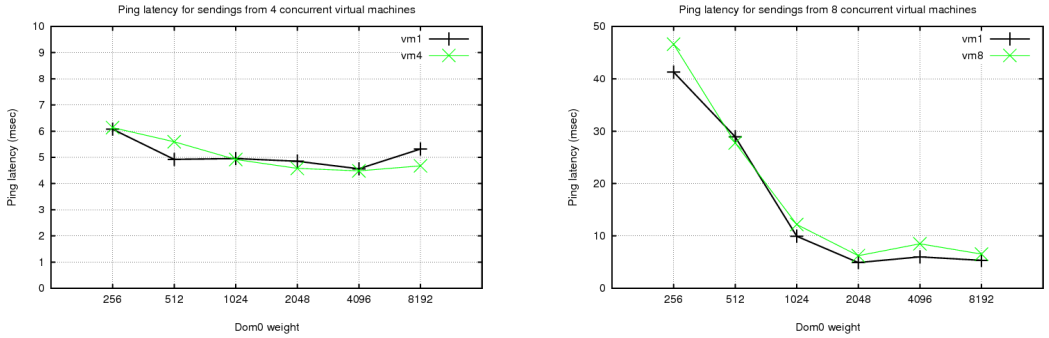


FIG. 26 – Architectures de test pour mesurer la latence de ping.

architectures de test correspondantes sont représentées sur la figure 26(a) pour les requêtes de ping sur ou depuis un hôte virtuel cohabitant avec d'autres hôtes virtuels émettant ou recevant des flux TCP, et sur la figure 26(b) pour les expériences sur les routeurs virtuels.

Résultats : Emission. Les tests avec 4 ou 8 machines virtuelles cohabitant une même machine physique dont une émet une requête de ping et les 3 ou respectivement 7 autres machines virtuelles émettent des flux TCP avec iperf sont représentés sur la figure 27. Chaque courbe représente une expérience. Sur chaque graphe, la courbe vm1 représente la latence du ping dans une expérience où le ping a été émis depuis la première machine virtuelle, la courbe vm4 ou respectivement vm8 représente la latence de ping dans une expérience où le ping a été émis depuis la dernière machine virtuelle. Ces différentes expériences ont été effectuées pour évaluer l'efficacité et l'équité. On observe que la latence ne varie pas, que le ping soit lance depuis la première ou la dernière machine virtuelle. L'augmentation du poids en domaine 0 contribue à diminuer la latence, notamment à partir d'un poids de 1024 pour 8 machines virtuelles.

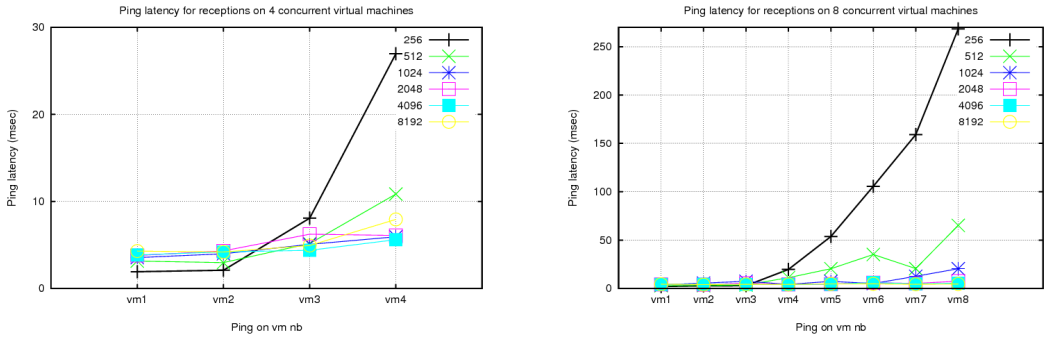
Résultats : Réception. Plusieurs expériences ont été effectuées en chargeant la machine de test par des flux TCP. Afin d'évaluer en particulier l'équité, l'expérience a été répétée dans chaque cas (4 ou 8 machines virtuelles) autant de fois qu'il y ait de machines virtuelles dans la machine de test. Dans chaque expérience, la requête de ping a été reçue sur une machine virtuelle différente. On observe ici à nouveau le phénomème de l'inéquité. En fonction du numéro de la machine sur laquelle la requête de ping est reçue, la latence varie. Elle augmente quand le numéro de machine virtuelle de la machine de test décroît. Sur les graphs (a) et (b) de la figure 28, chaque abscisse vm$X$ représente une expérience (ping reçu sur la machine



(a) (b)

FIG. 27 – Latence de ping sur une machine virtuelle parmi 4 (a) et 8 (b) autres machines virtuelles envoyant des flux TCP avec iperf.

virtuelle X, les autres machines virtuelles recevant des flux TCP). Chaque courbe donne les résultats pour un certain poids en domaine 0.



(a) (b)

FIG. 28 – Latence de ping sur une machine virtuelle parmi 4 (a) et 8 (b) autres machines virtuelles recevant des flux TCP avec iperf.

Résultats : Transmission. La latence de ping en transmission sur un routeur virtuel parmi d'autres routeurs virtuels qui sont chargés de transmettre des flux TCP décroît quand on augmente le poids du domaine 0. Les résultats pour 8 routeurs virtuels cohabitant une même machine physique sont représentées sur la figure 29.

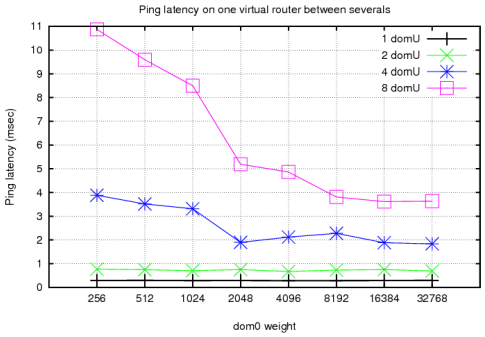


FIG. 29 – Latence de ping sur une machine virtuelle parmi 1, 2, 4 ou 8.

4.2.3 Analyse.

L'augmentation du poids du domaine 0 conduit à améliorer d'une part la latence de ping dans tous les cas (sur une machine chargée par des émissions/réceptions TCP ou un routeur), mais aussi l'équité dans le cas de la réception.
On observe que la latence de ping peut être améliorée en prolongeant la période d'ordonnancement du domaine 0.

5 Travaux relatifs

5.1 Etude des performances réseau avec XEN

Différentes études ont été menées auparavant pour évaluer les performances réseau offertes par XEN. La virtualisation des routeurs est un sujet émergeant. Il y a pour l'instant peu d'états de l'art qui étudie l'application de routeurs virtuels, par exemple à des fins de séparation de trafic comme le proposent McIlroy et Sventek dans [11]. L'évaluation de la virtualisation pour implémenter des routeurs virtuels est très récente et plusieurs articles ont été publiés pendant la période de ce stage.

XEN émerge en 2003, à l'origine comme solution de virtualisation pour les serveurs. Ses performances en tant que serveur web sont comparées [2] à celles de VMware workstation [3] et de User-Mode Linux [22] sur réseau local Gigabit Ethernet. L'ensemble des résultats en bande passante aggrégée des clients sur une série de requêtes est bien supérieur à celui de VMware et de User-Mode Linux. Ces premiers résultats rendent XEN intéressants pour l'utilisation dans le réseau.
Une étude détaillée des performances de XEN [12] analyse entre autre le comportement réseau dans les domaines invités et compare le débit TCP dans le domaine 0 au débit TCP dans les domaines U. Les résultats montrent que sur un seul domaine invité le débit TCP

peut atteindre la valeur maximale pouvant être offerte par un lien Gigabit Ethernet, soit 941Mb/s, ce que nous avons mesuré aussi sur des machines à deux CPU. Dans l'étude, des test sont également faits sur des machines à un seul CPU à titre de comparaison et pour un domaine U, la performance réseau chute d'environ 10% dans cette configuration. Ce surcoût est lié à l'hyperviseur qui doit exécuter des instructions supplémentaires pour gérer les domaines U et au filtrage et pontage en domaine 0 du trafic provenant de ou allant vers les domaines U. Des mesures de débit en domaine 0 montrent qu'à partir de 3 interfaces réseau utilisés en parallèle, le débit diminue d'environ 37% par rapport au débit sur un système linux sans virtualisation. Nous avons constaté une baisse de débit comparable en transmission sur des machines à deux interfaces réseau, mais dans le contexte où le trafic à été routé entre ces deux interfaces ce qui ajoute un coût supplémentaire.
Une étude similaire [23] mesure les performances réseau pour un trafic généré par iperf en émission depuis une machine virtuelle vers une machine physique distante. Deux configurations différentes sont utilisées. Dans la première expérience, la machine virtuelle émettrice et le domaine 0 utilisent le même CPU. Dans la deuxième expérience, le domaine 0 et la machine virtuelle émettrice utilisent des CPU différents. Les résultats montrent que si un seul CPU est utilisé par le domaine 0 et le domaine U, le débit TCP est dégrade d'environ 50% par rapport au débit TCP qu'on obtient sur un système linux sans virtualisation. Si le domaine 0 et le domaine U utilisent des CPU distincts, le surcoût de la virtualisation devient négligeable et le débit atteint une valeur très proche de celle obtenue sur un système linux sans virtualisation. La performance réseau dépend donc de la capacité de CPU de la machine. Dans nos expériences, pour une communication TCP saturant le lien sur un seul domaine U, au moins 50% du temps des deux CPU sont utilisés ce qui signifie qu'un seul CPU serait déjà saturé par la communication d'un seul domaine U ce qui correspond aux résultats décrits. Les auteurs ont mené une analyse plus détaillée du code de XEN qui montre que 60 à 70% du code est responsable pour la gestion des tables de pages mémoire et des *grant tables* pour la communication inter-machines virtuelles ce qui sature rapidement le CPU. Les commutations de contexte augmentent de 92% par rapport à linux hors contexte virtualisé et les pertes de TLB sont très fréquentes.
Une optimisation des mécanismes de gestion de pages mémoire et des interfaces réseau virtuelles dans XEN [24] diminue ce surcoût. Le mécanisme de mapping de pages est remplacé par des copies qui sont moins coûteuses. L'implémentation de superpages et d'une table de mapping de pages globale est aussi mise en œuvre pour réduire le nombre de pertes TLB. D'autre part, le fait que les interfaces réseau virtuelles sont beaucoup plus simples que les interfaces réseaux physiques implique que des fonctionnalités offertes par les cartes réseau comme le *TCP segmentation offloading*, le *scatter-gather I/O* et le *TCP checksum offloading* ne peuvent pas être utilisées par les machines virtuelles. L'ajout de ces fonctionnalités dans les interfaces virtuelles contribue à l'augmentation des performances réseau.
Après ces optimisations, une nouvelle évaluation de XEN [13] en transmission réseau, dont le but est d'étudier la virtualisation pour les routeurs montre que le débit d'une transmission sur un domaine U est inférieur à celui obtenu dans le domaine 0, tout comme les expériences précédentes. L'utilisation d'une machine physique plus puissante comportant deux

CPU améliore de manière significative ces performances. Avec plusieurs machines virtuelles transmettant du trafic simultanément, le débit cumulé de toutes ces machines diminue quand le nombre de machines virtuelles augmente, ce que nos expériences ont montré également. L'étude montre que le goulot d'étranglement est d'une part l'hyperviseur car l'utilisation de plusieurs machines virtuelles demande des commutations de contexte supplémentaires, et d'autre part le CPU qui est rapidement saturé par les transmissions réseau, or nous avons vu que pour les communications sur jusqu'à 8 machines virtuelles, deux CPU ne sont pas tout à fait saturés, mais le temps CPU disponible est mal réparti allant jusqu'à saturer le domaine 0.
Une étude sur le surcoût engendré par la virtualisation avec XEN [25] montre que le débit TCP n'est que faiblement affecté sur une machine virtuelle, même si celle-ci effectue des calculs supplémentaires. L'utilisation de plusieurs machines virtuelles simultanément pour des émissions ou réceptions réseau a pour conséquence une dégradation du débit TCP cumulé et une augmentation de l'inéquité de débit inter-flux. Nous avons étudié de plus près cette inégalité et observé qu'elle apparaît en particulier en réception de flux TCP. En plus, l'utilisation de CPU est très importante lors de tranmissions réseau et le temps CPU dédié aux transmissions réseau est très réduit lorsque la machine exécute des calculs supplémentaires. Des études plus récentes décrites dans la section suivante ont étudié l'impact de l'ordonnancement des machines virtuelles sur les communications réseau.

5.2 Etudes des ordonnanceurs dans XEN

Une comparaison des trois ordonnanceurs courants de XEN [14] qui effectuent un partage proportionnel du temps CPU a été effectuée. Ces ordonnanceurs sont le BVT scheduler (Borrowed Virtual Time), le SEDF scheduler (Simple Earliest Deadline First) et le Credit scheduler. Le partage proportionnel signifie que le CPU est partagé parmi les domaines invités actifs en fonction de leur poids. Le Credit scheduler est le premier à supporter les systèmes SMP et l'étude de comparaison montre qu'il permet le mieux d'exploiter les CPU sur les machines multiprocesseurs. Ceci est un facteur important car les études sur les performances réseau montrent que l'utilisation d'au moins deux CPU donne des résultats beaucoup plus proches des systèmes linux sans virtualisation. Nous avons constaté qu'avec le Credit scheduler, en effet, la charge est répartie sur les processeurs de manière à pouvoir exploiter pleinement les deux CPU. Parcontre, le fait d'avoir toujours deux domaines impliqués dans une communication réseau depuis un domaine U, le domaine 0 et ce domaine U, il serait une raison de ne pas partager chaque CPU parmi tous les domaines invités mais de dédier par exemple un CPU au domaine 0 et l'autre aux domaines U afin d'éviter des commutations de contexte trop fréquents et de permettre un travail en parrallèle du domaine 0 et des autres domaines.
Une étude très récente sur le comportement réseau dans le cas de plusieurs machines virtuelles [10] montre que la répartition de la bande passante sur les différents domaines U n'est pas équitable dans la configuration par défaut de la version courante de XEN, avec le Credit scheduler ce qui correspond aussi à nos mesures en réception TCP. La bande passante obtenue sur les différentes machines virtuelles décroît en fonction du numéro de

la machine virtuelle (les numéros sont attribués dans l'ordre croissant lors de la création des machines virtuelles). Les auteurs proposent des améliorations au niveau du canal de notifications d'événements qui ont permis un meilleur équilibrage inter-machines virtuelles de la bande passante, mais diminuent légèrement la bande passante agrégée des machines virtuelles. Nous avons essayé de contourner dans un premier temps le problème de façon expérimentale en attribuant plus de ressources au domaine 0 afin que celui-ci ait le temps de traiter tous les événements, ce qui a permis d'obtenir une équité à 20Mb/s près sur un lien à 1Gb/s.

6 Conclusion et perspectives

La virtualisation du réseau introduit une surcharge dans le traitement des communications dûe à une copie supplémentaire des paquets entre le domaine 0 et les domaines U et au multiplexage par pont ou routeur entre les interfaces virtuelles et physiques ce qui introduit un surcoût en CPU important en domaine 0. D'autre part, la virtualisation du réseau nécessite un partage combiné des ressources tel que les processeurs et les interfaces réseau. Nous avons montré que le partage du CPU est le point le plus sensible dans les communications réseau car tous les domaines invités ne sont pas impliqués de la même façon. Pour une communication réseau en domaine U, le domaine 0 est toujours impliqué. Nous avons étudié l'ordonnanceur et montré que l'allocation d'une part de CPU plus importante au domaine 0 qu'aux domaines U peut améliorer les performances réseau et l'équité.

XEN est en constante évolution et un de nos prochains objectifs est d'évaluer les performances réseau dans sa plus récente version, XEN 3.2 afin de déterminer l'impact des changements qui ont été apportés aux mécanismes de notification d'événements sur l'équité des flux réseaux inter-machines virtuelles. Même si notre paramétrage de l'ordonnanceur a permis d'augmenter le débit, nous voudrions également évaluer la latence et la gigue dans une expérience suivante. En particulier, nous voudrions évaluer la qualité de service de bout en bout pouvant être offerte entre machines virtuelles en implémentant des règles de différentiation de trafic.
Une adaptation automatique des paramètres de l'ordonnanceur au nombre de machines virtuelles en communication permettrait de rendre le modèle plus dynamique.

Il serait également intéressant d'étudier l'impact d'optimisations matérielles. Il est connu que la virtualisation a un surcoût spécialement élevé pour les opérations d'entrée/sortie. En utilisant des interfaces réseau programmables, on pourrait charger des fonctionnalités du niveau système d'exploitation au sein de l'interface même pour alléger le traitement en domaine 0.
Notre prochain objectif est d'intégrer le modèle de routeur virtuel dans le réseau de clusters virtuels de HIPCAL avec l'outil d'émulation de réseaux eWAN [26]. Les routeurs virtuels peuvent contrôler l'attribution des ressources aux différents clusters et les isoler au cœur-même du réseau.
Puis nous voudrions utiliser les routeurs pour implémenter différentes stratégies de routage

au sein d'un même routeur physique pour créer des réseaux virtuels de types différents qui cohabitent dans un même réseau physique. A long terme, nous prévoyons un déploiement du modèle à large échelle dans Grid'5000 pour permettre la cohabitation de plusieurs expériences avec des réseaux isolés.

Références

[1] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," in *SOSP '73 : Proceedings of the fourth ACM symposium on Operating system principles*, (New York, NY, USA), p. 121, ACM, 1973.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *SOSP '03 : Proceedings of the nineteenth ACM symposium on Operating systems principles*, (New York, NY, USA), pp. 164–177, ACM, 2003.

[3] J. Sugerman, G. Venkitachalam, and B.-H. Lim, "Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor," in *Proc. 2001 Usenix Annual Technical Conference*, pp. 1–14, Usenix Assoc., 2001.

[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] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *ACM Transaction on Computer Systems*, Sept. 1984.

[6] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*. Prentice Hall, 2007.

[7] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard, "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*, (Washington, DC, USA), pp. 99–106, IEEE Computer Society, 2005.

[8] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "iperf : testing the limits of your network," *http ://dast.nlanr.net/Projects/Iperf*.

[9] http ://wiki.xensource.com/xenwiki/ CreditScheduler.

[10] D. Ongaro, A. L. Cox, and S. Rixner, "Scheduling i/o in virtual machine monitors," in *VEE '08 : Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, (New York, NY, USA), pp. 1–10, ACM, 2008.

[11] R. McIlroy and J. Sventek, "Resource virtualisation of network routers," in *HPSR 06 : 2006 Workshop on High Performance Switching and Routing*, June 2006.

[12] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, "Diagnosing performance overheads in the xen virtual machine environment," in *VEE '05 : Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, (New York, NY, USA), pp. 13–23, ACM, 2005.

[13] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating Xen for Router Virtualization," in *ICCCN*, pp. 1256–1261, 2007.

[14] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three cpu schedulers in xen," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, 2007.

[15] N. Niebert, I. E. Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs, "Network virtualization : A viable path towards the future internet," *Wirel. Pers. Commun.*, vol. 45, no. 4, pp. 511–520, 2008.

[16] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox : Towards an operating system for networks," 2008.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. M. Parulkar, L. L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow : enabling innovation in campus networks.," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[18] L. Bonnet, "Etat de l'art des solutions libres de virtualisation pour une petite entreprise,"

[19] D. Gupta, R. Gardner, and L. Cherkasova, "Xenmon : Qos monitoring and performance profiling tool," *Technical Report HPL-2005-187, HP Labs*, 2005.

[20] E. Ackaouy, "[xen-devel] new cpu scheduler w/ smp load balancer - xen source," *http ://lists.xensource.com/archives/html/xen-devel/2006-05/msg01315.html*, 2006.

[21] "Creditscheduler - xen wiki," *http ://wiki.xensource.com/xenwiki/CreditScheduler*, 2007.

[22] H. jorg, H. Oxer, H. Hoxer, K. Buchacker, and V. Sieh, "Implementing a user mode linux with minimal changes from original kernel," 2002.

[23] P. Apparao, S. Makineni, and D. Newell, "Characterization of network processing overheads in xen," in *VTDC '06 : Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, (Washington, DC, USA), p. 2, IEEE Computer Society, 2006.

[24] 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*, (Berkeley, CA, USA), pp. 2–2, USENIX Association, 2006.

[25] P. Primet, O. Mornard, and J.-P. Gelas, "Evaluation des performances réseau dans le contexte de la virtualisation XEN," in *CFIP 2008 : Colloque Francophone sur l'Ingénierie des Protocoles*, (Les Arcs, France), Mar. 2008.

[26] P. Vicat-Blanc, O. Glück, C. Otal, and F. Echantillac, "Emulation d'un nuage réseau de grilles de calcul : ewan," Research Report RR2004-59, LIP, ENS Lyon, Lyon, France, Dec. 2004.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399