Network Virtualization - A Virtual Router Performance Evaluation

Bruno Parreira*†, Márcio Melo*†, João Soares*†, Jorge Carapinha*
*Portugal Telecom Inovação
Aveiro, Portugal
{bruno-m-parreira,joao-m-soares,marcio-d-melo,jorgec}@ptinovacao.pt

Romeu Monteiro†, Susana Sargento†
†Instituto de Telecomunicações
University of Aveiro
Aveiro, Portugal
{romeumonteiro7,susana}@ua.pt

***Abstract*—Network Virtualization is claimed to be a key component of the Future Internet by enabling the coexistence of heterogeneous (virtual) networks on the same physical infrastructure [1], providing the dynamic creation and support of different networks with different paradigms and mechanisms. In order for virtualization to be used in a network operator's infrastructure, its impact on the network traffic must be studied.**

In this paper, we perform an analysis of the impact of network virtualization on two types of traffic, TCP and UDP. To deploy the virtual networks, the Network Virtualization System Suite is used. This platform enables the creation of virtual networks on top of a substrate network, isolating the traffic in the different layers. The tests performed evaluate the effect that the increase of virtual routers and data flows has on throughput and packet delay. The effect of CPU load on throughput is also analyzed.

The results obtained using TCP demonstrate that the CPU load has a more adverse effect on throughput than increasing the number of virtual routers, with a loss of 25% in the first case and 15% in latter case. The UDP tests revealed that increasing virtual routers leads to an increase in packet delay variation.

***Index Terms*—Network Virtualization, Virtual Router, Virtual Network, Network Performance.**

I. Introduction

In the last few years the Internet has been walking steadily towards the Networks of the Future. These necessary changes still face a lot of resistance from legacy networks, which are based on technologies designed decades ago. Current networks lack the dynamism and the flexibility necessary for these changes to take place. Cloud Computing can be seen as an example of a paradigm being hindered by current network infrastructures. For example, a company moving its Information Technology (IT) resources to the cloud will probably use Virtual Private Networks (VPNs) based solutions to connect virtual infrastructures with their premises. VPNs were not designed to adapt to the users demand, a characteristic that is very popular with Cloud Computing [2].

Network virtualization can play an important role in the development of the Networks of the Future. It brings great improvements in terms of flexibility, isolation and dynamism, which will foster the development of new architectures and technologies while improving current network based services [3].

Currently there are various alternatives to deploy virtual networks, with one of them being the Network Virtualization System Suite (NVSS) [4] developed under the 4WARD project [5]. The NVSS is a platform for the creation, discovery, monitoring and management of virtual networks; it will be the one used in the tests performed in this paper. Although some tests have proven the functional capabilities of the platform, see [6] and [7], a data quantitative analysis is still missing. Several web based services like video streaming or voice calls have minimum requirements, in terms of throughput or jitter, that need to be met for these services to be deployed. This paper will cover performance parameters like throughput and packet delay that have a direct impact on the provisioning of services.

To perform these tests two types of traffic will be generated, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The results obtained using TCP demonstrate that the Central Processing Unit (CPU) load has a more adverse effect on throughput than increasing the number of virtual routers, with a loss of 25% in the first case and 15% in latter case. The UDP tests revealed that increasing virtual routers leads to an increase in packet delay variation.

The rest of the paper is organized as in the following. After summarizing the related work in section II, section III describes the architecture of the NVSS platform and presents the existing functionalities. Section IV describes the testbed used and evaluates the influence on the packet delay variance with the number of virtual routers and network flows. Section V analyses the virtual router throughput and investigates the influence of the CPU load on the overall throughput, and section VI concludes the paper and describes the future work.

II. Related Work

Future Internet research projects such as the PlanetLab [8] or the 4WARD [5] have investigated and promoted the use of network virtualization as a way to evaluate and deploy, in the latter, future Internet architectures.

With that in mind, Egi et. al [9] evaluated the use of Xen hypervisor [10] as a way to implement virtual routers. The performance of virtual routers on commodity hardware has been assessed in [11] and [12], where the virtual routers throughput is similar to the one of underlying hardware when using control plane from forwarding plane separation.

A platform for high performance and flexible virtual routers on commodity hardware based on multiple input queues has been proposed in [13]. The virtual router migration feature has proposed by Wang et. al [14] as a primitive of network management operations. To reduce the Virtual Router (VR) downtime due to the migration process, Wang et. al [15] proposed the separation of the forwarding plane from the control plane.

Despite the existing performance research results provided in [11] and [12] on virtual routers running on commodity hardware, we argue that a deeper analysis on either the traffic types or the influence on the CPU load is still not tackled. We also argue that the impact on the packet delay variance with the number of virtual routers and without using control and forwarding plane separation is not assessed.

In the following section we present and describe the NVSS architecture and its built-in functionalities to handle virtual networks.

III. Network Virtualization Architecture

The goal of the developed virtualization platform is to provide the operators with a network virtualization solution that is easy to use, versatile, and efficient in virtual network discovery. The resulting platform provides the necessary functionalities to discover, monitor, deploy and manage virtual networks running on top of a substrate network. It is designed to run on Fedora Core 8 and Debian Lenny Linux distributions with the Xen kernel. Figure 1 presents the network virtualization considered approach, which takes into account a heterogenous physical network and builds on top of it virtual networks with different types of topologies.

A. Network Virtualization System Suite Architecture

The NVSS is composed of 3 software modules: the Agent module, the Manager module and the Control Centre module; their hierarchical decomposition is demonstrated in Fig. 2. The Agent module is designed to work within the domain of a Xen virtualization environment, running on every substrate node, in order to perform network enforcements and periodically gather data. The Agents, besides interacting with each other to share network topology information, also receive and send requests to the Manager, which is a centralized entity in charge of aggregating all Agents' knowledge and sending them commands. The Manager is also devoted to map new Virtual Network (VN) requests and to communicate with the Control Centre, which is the user's front-end, and provides him with graphical and simple to use virtual network creation, management, and monitoring functionalities.

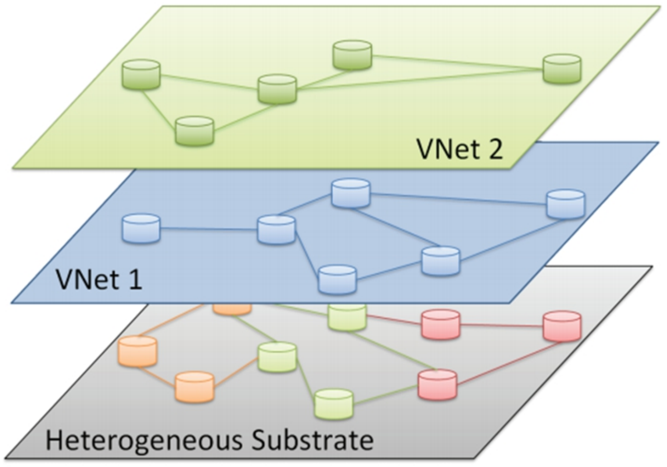

Figure 1. A Network Virtualization Approach

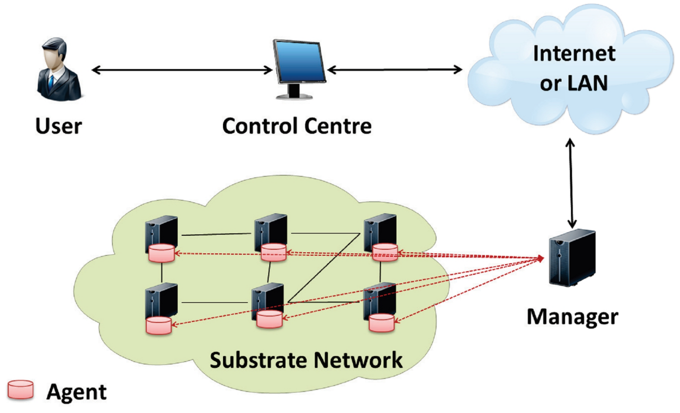

Figure 2. Network Virtualization System Suite - Architecture

B. Virtual Network Mapping and Creation

The Control Center module provides the user with means to create and embed a new VN. By selecting and placing resources on the platform Graphical User Interface (GUI) and by connecting them with links, a VN can be specified. The user may specify the resources' CPU capabilities, Random Access Memory (RAM) amount, location, number of interfaces and also perform network addressing configurations. The final step in creating a new virtual network is to commit it to the Manager, which will then map it in the physical infrastructure. The embedding problem, that includes both nodes and links mapping, is a complex one and requires a trade-off between computation time and embedding optimization. In order to lower the computational requirements, a heuristic mapping algorithm was developed, which aims to embed VNs taking into consideration both the substrate links' and nodes' loads.

C. Substrate and Virtual Network Monitoring

Dynamic resource monitoring is fundamental to provide an accurate view of the virtual and physical networks, and to quickly react to failures or configuration problems. The implemented monitoring functions periodically update the resources' information; therefore it is possible to quickly identify diverse situations, such as failures and high resource usage. Every Agent periodically checks its local resources' configuration and status, and reports back to the Manager if any change occurs. Several parameters are monitored: CPU load, RAM, Hard Disk Drive (HDD) usage, interface and link status, interface bridge attachment and configuration, number of running virtual machines and their state.

D. *Virtual Network Management*

Just like the previously described monitoring ability, the management feature is also a crucial one; to that end, some functionalities are provided. It is possible to change the resource's state, i.e.: reboot, shutdown, suspend or power up; to change the assigned RAM memory in runtime; and to delete either a single resource or a complete VN, which greatly simplifies the administrator work.

IV. Evaluation - Packet Delay

In this section we start with the description of the testbed considered to evaluate several packet delay statistics as functions of the number of virtual routers and the number of network flows. The statistics considered are average packet delay, packet delay variance and packet delay variation.

A. *Testbed Configuration*

In order to analyze the impact of network virtualization on the packets' delay times, several aspects must be taken in consideration. For example, it must be guaranteed separation between the several flows of traffic, regardless of whether they traverse the same virtual router or not. Another important consideration is to guarantee that all traffic traversing the virtual routers is captured and limited to the one injected into the network. With this in mind, a testbed was designed, which is presented in this section.

The testbed used is composed of six computers using the configuration shown in figure 3. Three computers are used to generate the traffic flows, two computers are used to receive them, and one computer is used to deploy the virtual routers. The 5 computers used to generate and receive the traffic flows provided us with 14 ethernet interfaces, thus allowing for a maximum of 7 flows at a time, since we want to observe independent flows which don't start nor end at the same interfaces. Since there could be no more than 7 flows, the number of VRs was also limited to a maximum of 7. The computer, Eddie, where the virtual routers are instantiated, is an Intel Xeon E3220 with four cores (2.4GHz each) and 6GB of system memory. This computer runs Xen Hypervisor version 3.1. The machines responsible for traffic generation are connected to a switch which sends all incoming traffic to one port. This port is connected to a hub ensuring that the same packets which enter the virtual routers are captured by another machine using the *Wireshark* software [16]. On the right side of the virtual routers, a symmetric configuration is used.

To ensure independency between the different virtual links, Virtual Local Area Network (VLAN) tagging was used; VLAN tagging is only applied between the switches. In the physical machine Eddie, where the VRs reside, each virtual interface is associated with a specific bridge and VLAN tag.

Iperf [17] sessions are used to generate traffic flows with a packet size of 1300 bytes at a bit rate of 1Mbps. The data

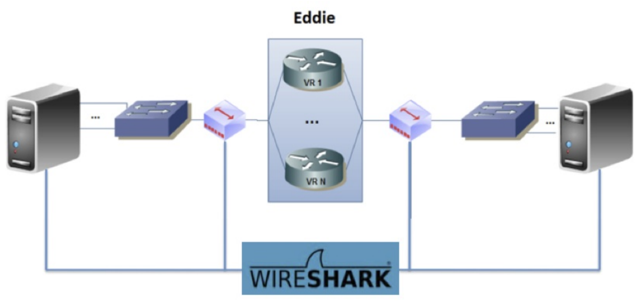

Figure 3. Experimental Apparatus

obtained in each test derive from 10 runs of 30s of traffic. To have an *end2end* view of the packet transmission, the *Wireshark* software was used in four different points along the path:

1) In the machine generating the flow;
2) In the hub before Eddie;
3) In the hub after Eddie;
4) And in the machine receiving the flow.

The *Wireshark* application uses the machine's system time to get the capture time of each packet.

The results obtained for packet delay are based on the time it takes for packets to go from hub1 to hub2, so as to evaluate the delay introduced by the virtual routers in Eddie. The timestamps of the packets were collected using the *Wireshark* software, and the data inside these packets was used to match the timestamps at the 2 hubs. This way, it was possible to obtain the individual delays for each packet in each flow.

To determine the packet delay variation, it was used the method defined in [18]. This method defines packet delay variation as the difference in delay times between two consecutive packets.

B. *Results*

In figures 4 and 5 we can observe the average packet delay for a set of 15 runs with confidence intervals of 90%. Figure 4 shows the effect of changing the number of active VRs while keeping constant the number of flows per VR; figure 5 shows the effect of increasing the flows in a single VR.

It is visible that the average packet delay is very stable regardless of the number of active VRs and flows per VR. The very small confidence intervals corroborate this stability among runs. This is likely due to the low amount of extra load imposed on the system by the extra traffic and extra VRs, which might lead to an average delay change that is small compared to the total average delay.

In figures 6 and 7 we can observe the variance of the packet delay for a set of 15 runs with confidence intervals of 90%. Figure 6 shows the effect of changing the number of active VRs while keeping constant the number of flows per VR, while figure 7 shows the effect of increasing the flows in a single VR.

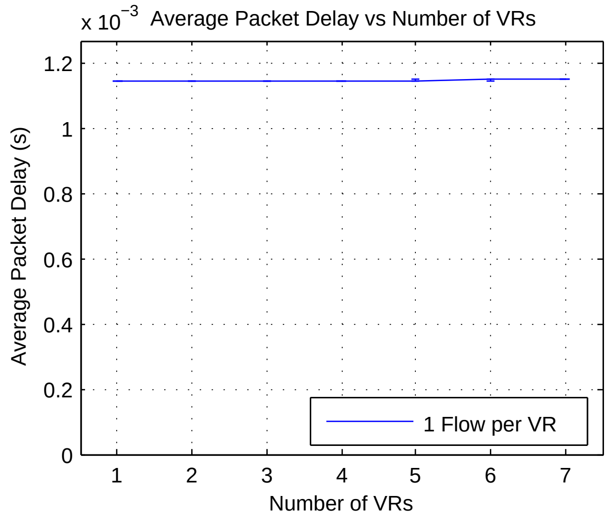

Figure 4. Average packet delay for a single flow per active VR and different numbers of VRs

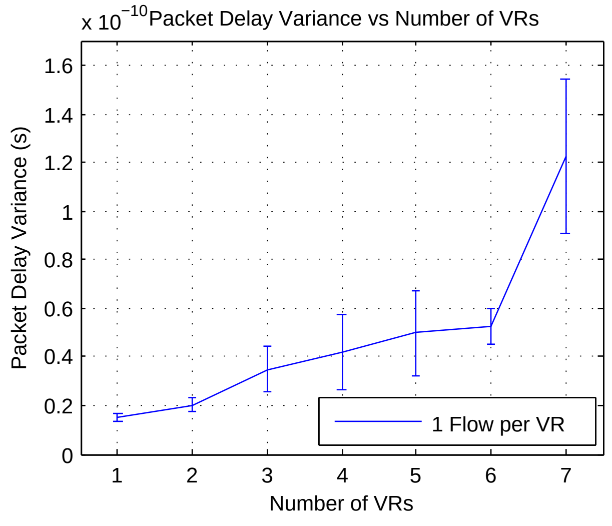

Figure 6. Packet delay variance for a single flow per active router for different numbers of VRs

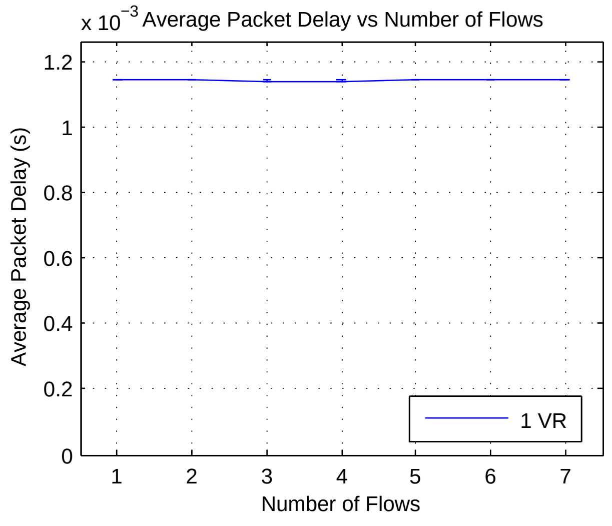

Figure 5. Average packet delay for a single active VR with different numbers of flows

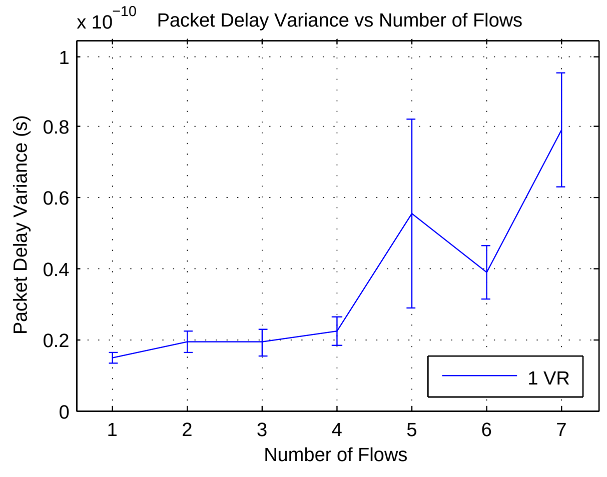

Figure 7. Packet delay variance for a single active VR with different numbers of flows

We can see that the general trend is that the variance of the packet delay, as well as the confidence intervals, increase with the number of active VRs and flows per VR, with a few exceptions. This means that, as the number of active VR and flows per VR grow, the delays become more statistically unstable on 2nd order time measures: while the average packet delay remains constant, the variance increases on average but becomes more unstable between runs.

In figures 8 and 9, it is possible to observe the behavior of the average packet delay, as well as the packet variance, when different combinations of VRs and flows per VR are activated up to a combined maximum of 7 packet flows.

For the average packet delay in Figure 8, we can see that it is similar regardless of the number of VRs and flows per VR, always close to 1.15 ms.

When we observe the variance of the delays of the packets in Figure 9, we can see very clearly that the larger the number of active VRs and the larger the number of flows per VR, the larger is the variance of the packet delay. For the same cases analyzed in Figures 8 and 9, we also studied the throughtput behavior. For all these cases we obtained values of throughput always averaging very close to 100% with very small confidence intervals. This is expectable due to the small bandwidth used by each flow (1Mbps) which, even when considering the maximum combination of 7 flows, is still too low to be affected.

In figure 10 it is represented the packet delay variation while varying the number of active routers.

The packet delay variation is similar for 1 and 2 VRs; for

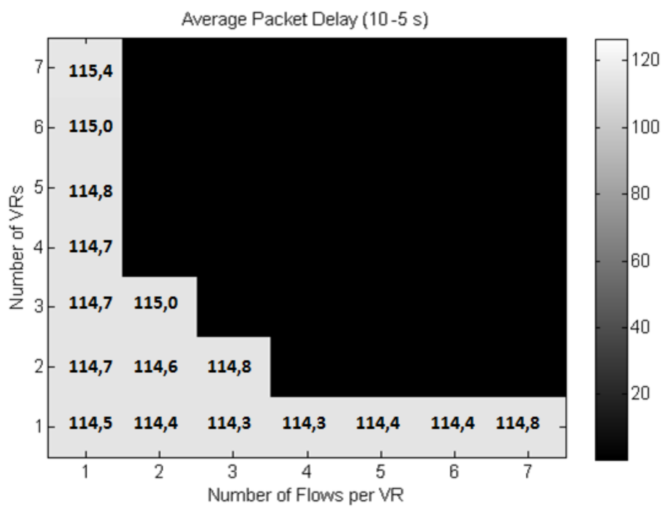

Figure 8. Average packet delay for a maximum total of 7 flows

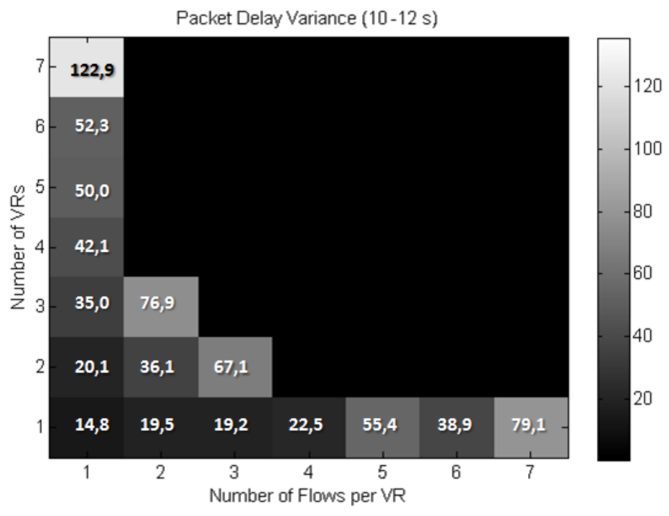

Figure 9. Packet delay variance for a maximum total of 7 flows

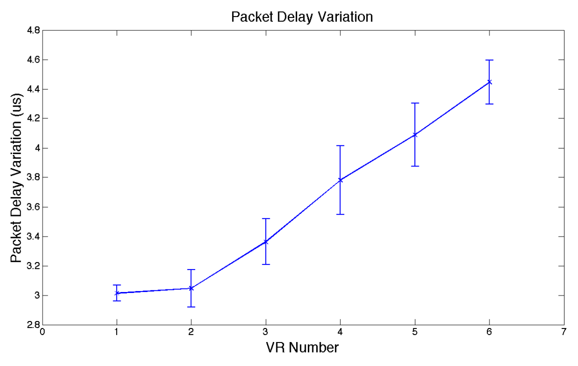

Figure 10. Packet Delay Variation while varying the number of VRs

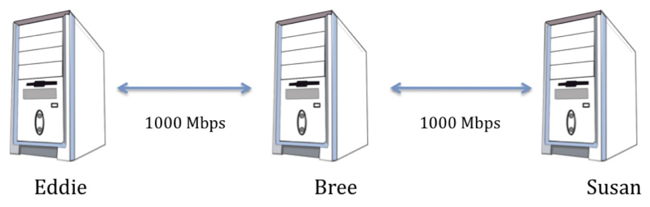

Figure 11. Experimental Apparatus - Throughput

2 or more VRs, it increases almost linearly with the number of VRs.

V. Evaluation - Throughput

In this section we start with the description of the testbed considered to evaluate the network throughput, and furthermore, we assess and analyze the network throughput as a function of the number of virtual routers, the number of network flows, and also as a function of the CPU load.

A. *Testbed Configuration*

The purpose of these tests is to analyze the behavior of the throughput with a different number of virtual routers and CPU load. The changes made to the configuration of the testbed are due to the fact that the ports on the hubs used in the previous setup are limited to 10 Mbps.

In this configuration, only three machines were used (see figure 11: Eddie as a transmitter; Susan as a receiver; and Bree where the VRs were mounted. Bree has an Intel Xeon E3110 with two CPU cores (3.0 GHz each) and 6GB of system memory.

The three nodes are directly connected through Ethernet cables with a bandwidth of 1000Mbps. To test the throughput the software *Iperf* was used, which will allow the measuring of the transmission rate between Eddie and Susan. In all the tests, 15 runs of 30 seconds of traffic were analyzed. The traffic is composed of TCP packets with a fixed size of 16 KB. To separate the different flows of traffic, VLAN tagging was used.

B. *Results - Throughput*

In order to establish a reference value, the throughput between Eddie and Susan was first measured without the use of VRs. This reference serves as a base comparison for the upcoming results. The throughput registered was constant and with a value of 940 Mbps.

In the first part of these tests, the throughput was measured while varying the number of active VRs; the number of VRs ranges from 2 to 7. The obtained results can be seen in figure 12 where the throughput shown is the combined value of all the flows. Assuming the reference value corresponds to 100% of the available throughput, the losses range from 13,5% to 4,9% (with 7 VRs and 4 VRs). Also, during these tests it was possible to observe that the hypervisor manages to make a fair distribution of resources. Looking at the flows' throughput individually, the values registered showed a small dispersion.

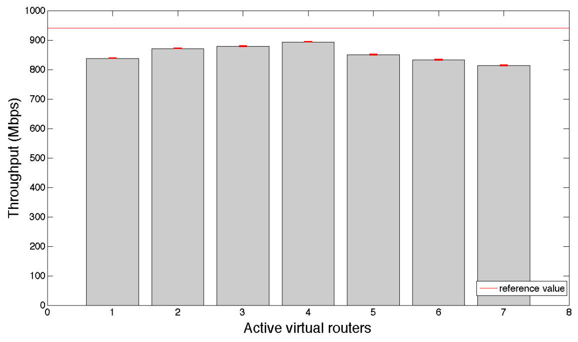

Figure 12. Throughput behavior while varying the number of VR

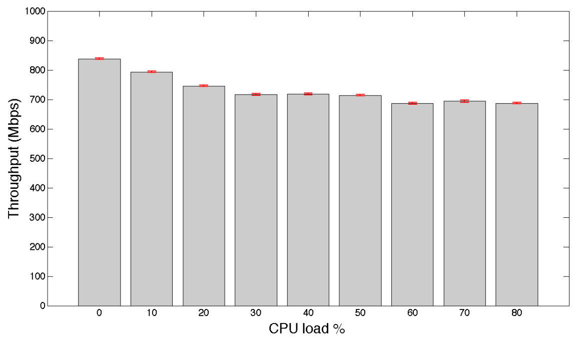

Figure 13. Throughput behavior while varying CPU load

To assess the influence of the CPU load in the VRs performance, a program called *Lookbusy* [19] was used. This software uses infinite loops in multiple threads to force the CPU load to a predetermined value. During this test only 1 VR was active.

The results can be seen in figure 13 with the confidence intervals of 90%. The results show that the amount of CPU load significantly affects the throughput. The throughput decreases steadily for CPU loads below 30%, while for higher loads the throughput seems to stabilize around 700 Mbps. Before making these tests, a trial run was made in which no virtual routers were used. During this trial run, the throughput seemed independent from the CPU load.

VI. Conclusion & Future Work

The main goal behind these tests was to establish a profile in terms of performance of network virtualization using the NVSS. The tests using TCP traffic allowed the retrieval of the throughput behavior while varying the number of virtual routers and CPU load. The tests with UDP traffic permitted an assessment of the behavior of packet delay in terms of average value and variance, as well as its variation, for different amounts of active routers and flows per router.

The results obtained for the throughput show that network virtualization has a negative impact on performance, as expected. The best results were obtained with 4 VRs, with the throughput presenting a loss of around 5%. Up to 4 VRs, the throughput increases with the number of active routers, which is probably due to a more efficient use of the bandwidth. Beyond this point, increasing the number of VRs decreases the global throughput. It should also be noted that having 7 VRs on the same machine never led the CPU load to exceed the 10% threshold, although it got close to this value. Looking at the results of the throughput behavior in terms of CPU load, at 10% of this load the throughput stays near 800Mbps. These two facts together lead us to believe that the throughput decrease after 4 VRs must be due to the increase of CPU load.

The measurements on the average packet delay and packet delay variance showed that, for different combinations on the number of VRs and flows per router, the average delay is mostly constant while the delay variance increases with both the number of VRs and the number of flows. We can also see that the confidence intervals for the variance also increase in the same way, which shows an increasing degree of instability for 2nd order parameters of the delay probabilistic distribution. Apart from the results for 1 VR, the packet delay variation increases almost linearly with the increase of VRs. In qualitative terms, the packet delay variation is low, even for the worst case where it reaches 0,4% of the packet delay. Also, looking at these results from an absolute value point of view, values in the order of microseconds indicate a good performance. Several types of data services which operate in real time, e.g. video or VoIP calls, can support a significant amount of average delay but need this delay to be stable. Therefore, one must make sure that the packet delay variance falls within certain limits. With this analysis, we analyzed how virtualization impacts this important aspect for real-time data services.

Looking at the combined results of the TCP and UDP traffic, the obvious conclusion is that up to 4 VRs network virtualization has minimum impact on throughput. Average packet delay is also minimally impacted, even though its variance and variation increase as the number of VRs and flows increases. Also, it seems that the CPU load should be the main factor behind the deterioration of the throughput performance. Even though using newer and better hardware should improve greatly the results obtained, given the age of the hardware used, the trend should remain that the more flows and VRs, the larger is the packet delay variance and jitter and the lower is the throughput.

There are several paths which can be trailed to proceed from this work. It would be important to study how the use of other virtualization technologies and hardware impacts the throughtput and packet delay statistics. On the other hand, it would also be relevant to have a direct comparison between the performances of virtualized and non-virtualized networks for the same kind of services. Experimenting with a larger number of VRs and network flows would allow us to have a more complete and detailed view of the impacts of virtualization; it would also allow us to relate the CPU load with the number of VRs, since for the number of VRs considered in this paper the CPU load was always low.

References

[1] N. Chowdhury and R. Boutaba, “Network virtualization: state of the art and research challenges,” *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20 –26, july 2009.

[2] B. Ahlgren, P. Aranda *et al.*, “Content, connectivity, and cloud: ingredients for the network of the future,” *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 62 –70, july 2011.

[3] N. Chowdhury and R. Boutaba, “Network virtualization: state of the art and research challenges,” *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20 –26, july 2009.

[4] J. Nogueira, M. Melo *et al.*, “Network virtualization system suite: Experimental network virtualization platform,” in *TridentCom 2011, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2011.

[5] 4WARD Consortium, “Virtualisation approach: Concept,” ICT-4WARD project, Deliverable D3.1.1, Tech. Rep., Sep. 2009.

[6] J. Nogueira, M. Melo *et al.*, “A distributed approach for virtual network discovery,” in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, dec. 2010, pp. 277 –282.

[7] ——, “Virtual network mapping into heterogeneous substrate networks,” in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 28 2011-july 1 2011, pp. 438 –444.

[8] L. Peterson and T. Roscoe, “The design principles of PlanetLab,” *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, p. 11–16, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1113361.1113367

[9] N. Egi, A. Greenhalgh *et al.*, “Evaluating xen for router virtualization,” in *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, 2007, p. 1256–1261. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4317993

[10] P. Barham, B. Dragovic *et al.*, “Xen and the art of virtualization,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, p. 164–177, 2003.

[11] S. Bhatia, M. Motiwala *et al.*, “Trellis: A platform for building flexible, fast virtual networks on commodity hardware,” in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008, p. 72. [Online]. Available: http://dl.acm.org/citation.cfm?id=1544084

[12] N. Egi, A. Greenhalgh *et al.*, “Towards high performance virtual routers on commodity hardware,” in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008, p. 20. [Online]. Available: http://dl.acm.org/citation.cfm?id=1544032

[13] ——, “A platform for high performance and flexible virtual routers on commodity hardware,” *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 127–128, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1672308.1672332

[14] Y. Wang, J. van der Merwe, and J. Rexford, “VROOM: virtual routers on the move,” in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, 2007.

[15] Y. Wang, E. Keller *et al.*, “Virtual routers on the move: live router migration as a network-management primitive,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, p. 231–242, Aug. 2008.

[16] “Wireshark · go deep.” http://www.wireshark.org/. [Online]. Available: http://www.wireshark.org/

[17] A. Tirumala, F. Qin *et al.*, “Iperf: The TCP/UDP bandwidth measurement tool,” *htt p://dast. nlanr. net/Projects*, 2005.

[18] “RFC 3393 - IP packet delay variation metric for IP performance metrics (IPPM),” http://tools.ietf.org/html/rfc3393. [Online]. Available: http://tools.ietf.org/html/rfc3393

[19] “Lookbusy – a synthetic load generator.” [Online]. Available: http://www.devin.com/lookbusy/