Network Virtualization: a Step Closer For Seamless Resource Mobility

Márcio Melo*†, Jorge Carapinha
*Portugal Telecom Inovação
Aveiro, Portugal
{marcio-d-melo,jorgec}@ptinovacao.pt

Susana Sargento
†Instituto de Telecomunicações
University of Aveiro
Aveiro, Portugal
susana@ua.pt

***Abstract*—Network Virtualization has gained prominent attention in the recent years, not only from the academia, but also from the Industry, mainly due to its numerous features such as flexibility, programmability, elasticity and dynamicity.**

One of the key assets with Network Virtualization is the ability to move components of the virtual network, or even the entire virtual network, from one or several physical hosts to others in real-time and seamlessly to the end-users.

This paper addresses virtual resource mobility from a new perspective: it proposes Virtual Network (VN) Clone migration, which requires no assumptions regarding the protocols running inside the virtual networks or its own architectures, leaving space for different types of protocols and architectures to be implemented, tested and used in production scenarios. The results are very promising: the VN Clone migration achieves no VN downtimes and it takes just a few seconds to be fully performed. This makes the VN Clone approach suitable both for non-real time traffic and voice over IP communications.

***Keywords*—Migration, Mobility, Network Virtualization, Virtual Networks, Virtual Router.**

I. Introduction

Network Virtualization has gained an increasing prominence in networking and telecommunication fields in the last few years. Initially, the interest in network virtualization was mainly pushed by Future Internet research initiatives [1]–[4], with the objective to find a platform on which novel Internet architectures could be experimented and evaluated without limitations or constraints, namely those associated with the traditional IP model.

One of the major features of Network Virtualization is the possibility to move virtual resources, i.e. Virtual Router (VR), on-demand and seamlessly from one physical host to another without losing network connectivity, therefore without affecting the IP layer. The VR migration process was initially proposed by Wang et al. [5], [6] as a primitive for network management tasks.

This paper focuses on the Virtual Network (VN) migration process[1] applied to the Network Virtualization, and it proposes VR Cloning as an alternative to VR live migration [5]. VR Cloning is based on cloning the VR, that is, on saving the current state of the VR and transfer the VR clone to the new physical host. This proposed approach achieves no VN downtime and it takes 2.75 seconds to be fully performed.

The rest of the paper is organized as follows. After summarizing the related work in section II, section III describes the virtual router migration process. Section IV evaluates the impact of the VN migration process either using the VR live migration or using the VR cloning migration. Finally, section V concludes the paper and describes the future work.

II. Related Work

The Virtual Machine (VM) migration was initially proposed for data-centers as a way to move the CPU load from one physical server to another. This feature is not only important for load balancing purposes, but it can also be applicable for planned maintenance operations, i.e. moving away all the VMs from one physical server that needs to be rebooted or shutdown for maintenance to different physical servers.

In order to reduce the downtime due to the VM migration process, Clark et al. [7] proposed the live migration of VMs, within the same LAN, which allows a VM to be migrated while still running. This not only reduces significantly the downtime provoked by the VM migration, but it also makes the migration process seamless to the end-users or to the running applications.

The VM migration approach is also applicable and important to the networking area, where it can be used for networking maintenance operations or for networking service deployment.

With that in mind, Wang et al. [5] proposed Virtual ROuters On the Move (VROOM) as a primitive for networking management tasks which makes it possible to move virtual routers freely without changing the IP-layer topology. Wang et al. [6] extended their prior research work on VR migration and proposed to decouple the data plane from the control plane of the VRs, which results in no performance impact on the data traffic when a hardware data plane is used, and very low impact when a software date plane is used.

Lo et al. [8] used the virtual router migration [6] as a primitive to perform the virtual network migration, i.e. the migration of an entire VN, and also proposed three algorithms to address the VN migration scheduling problem and to minimize the total migration cost.

Although the separation of the data plane from the control plane seems to be a very effective approach on the VR

[1]We consider the VN migration process as being the set of all actions required to perform the relocation of the VN or just parts of it (e.g. VR). We also consider the VR migration as the relocation of the VR only.

978-3-901882-50-0

migration, once it reduces the downtime of the VN, it is also a limiting factor on the conception of new network architectures and on the deployment of new network protocols. We argue that not only the VN migration process should be independent of the networking protocols that are running on the virtual network, but also no assumption should be taken on the router architecture itself. Therefore, we consider each VR as a black-box and propose the VR Cloning as an alternative to the current VR live migration process [5].

III. VN Clone Migration Procedure

The VN migration process can be started from a predicted event in the sense that it can be scheduled in time, i.e. planned maintenance, or it can be triggered from a non-predicted event, i.e. VN security attack or hardware failure. In this section, we describe the different actions required in our approach to perform the VN cloning.

The different operations which compose the VN clone migration process, with and without VR suspend, are described here:

1. $VLs\ Setup$ - This operation comprises the setup of new virtual links, e.g. setup of VLAN interfaces and virtual bridges, and it is considered as non-critical, since it is performed beforehand and in the time frame of the order of milliseconds (or even nanoseconds with optical switches).
2. $Clone/Move/Restore$ - This is the most critical and time consuming task to the overall VN migration process, and it can be divided into three sub-operations:
   a) $Clone\ VR$ - The cloning of the VR involves saving the current state of the memory RAM, e.g. routing tables, to the physical host hard-disk or even to a RAM-disk. This operation can be considered critical if the VR is put into suspend mode while being cloned, or non critical if the VR is still operating while being cloned. The time required to perform the VR cloning is given by formula (1), where VR_{memory} is the memory RAM size of the VR.
   b) $Move\ VR\ Clone$ - This action encompasses the transfer of the VR clone to the new physical host and it is, in principle, the most time consuming task of the three sub-operations, and of the overall VN clone migration process. Despite contributing to the VN migration execution time, it does not affect the VN downtime once the VN is still running while the VR clone is being relocated. The time required to perform this task is given by formula (2), where $BW_{reserved}$ is the bandwidth which is effectively reserved by the operator to this kind of operations, and BW_{free} is the bandwidth that is not provisioned (or available) on the physical path between the physical hosts and at that time period. Note that, if this phase takes too much time to be performed, the VR clone can easily become outdated and the re-transmission of control plane messages should take place (e.g. OSPF already supports re-transmission of control plane messages).
   c) $Restore\ VR\ Clone$ - The restore of the VR clone is performed after it is allocated on the new host, and it does not influence the VN downtime once the VR clone is not yet connected to the VN. This operation is performed in the time-frame of high hundreds of milliseconds.
3. $Add\ Virtual\ Bridge\ Int.\ \&\ Destroy\ Original\ VR$ - This is a critical action and encompasses adding a new virtual interface end-point to the virtual bridge that will be used to connect the VR clone to the VN. It also includes the shutdown of the original VR which is performed in order to avoid duplicated VRs operating on the virtual network. The VN transition, from the old VLs and VRs to the new ones, is signaled by the execution of these two operations. The VR shutdown (destroy) is executed in the time-frame of low hundreds of milliseconds.
4. $Remove\ VLs$ - The removal of old virtual links is a non-critical action and is performed in the same time-frame as the setup of virtual links (i.e. milliseconds). This phase does not count to the execution time, though it is part of the VN migration process.

$$t_{clone} = \frac{VR_{memory\_size}}{RamDisk_{write\_speed}} \quad (1)$$

$$t_{move} = \frac{VR_{memory\_size}}{BW_{reserved} + BW_{free}} \quad (2)$$

IV. VN Migration Evaluation

In this section, we describe the network virtualization testbed implemented to evaluate the proposed VN migration process. Two different VN migration methods are evaluated: the live migration and the clone migration with and without VR suspend. The evaluation metrics considered are the downtime and the migration execution time.

A. Testbed Scenario

The testbed is composed by 6 physical hosts and is depicted on the bottom of Fig. 1. The physical hosts are running Fedora Core 8 with Xen Hypervisor [9] version 3.1, and the virtual hosts are also running Fedora Core 8 with para-virtualization. Figure 1 also represents one VN, the VN $Vegas$, which was deployed using the Network Virtualization System Suite (NVSS) [10], and which is used to perform the VN migration process. All VRs are running the Quagga routing suite [11], and are using the OSPF as the routing protocol which is configured with the default parameters. In order to generate traffic across the virtual network $Vegas$, two physical machines were attached to it: one physical server that will be acting as a traffic generator, and one physical client which will consume the traffic. The physical links used by the server and the client are of 100 Mbps each.

The dashed lines on Fig. 1 represent the new location of the VR $Nick$ and the new location of the VLs. The VR $Nick$ is the one that will be triggered to be migrated from physical host $Mary$ to physical host $Bree$. The physical hosts, $Mary$ and $Bree$, are rack servers with CPU Intel® Xeon® X3330@2.66GHz and E3110@3.00GHz, respectively, configured with 6GB of RAM and both using Gigabit interfaces.

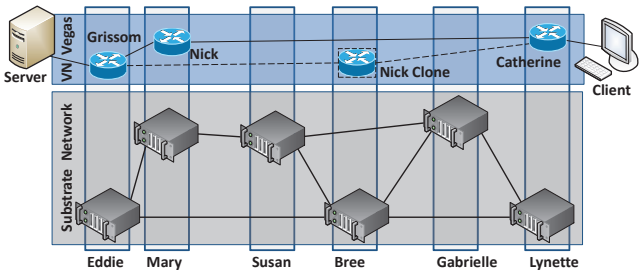


Figure 1. Network Virtualization Testbed: Virtual Router Migration Scenario.

B. Evaluation Results

To analyze the different VN migration methods, the traffic generator D-ITG [12] is used to evaluate the impact on the traffic carried by the virtual network due the VN migration process. The total experiment time is set to 100 seconds, where the traffic is continuously generated before, during and after the VR migration event. It is considered UDP traffic with a packet size of 1000 Bytes, and either with a bitrate of 10Mbps (i.e. 1250 packets/s) or of 20Mbps (i.e. 2500 packets/s) to evaluate the influence of the traffic bitrate on the VN migration process. The memory RAM of the VRs, i.e. the size of the routing tables, is set from 64MB to 256MB with intervals of 32MB. During each experiment, it is measured the number of packets sent and the number of packets lost. The execution time of each Linux command is also measured: bridge setup, VLAN setup, bridge interface setup, VR cloning, VR move, VR clone restore, VR destroy, VR live migration (only for the live migration process). For each experiment, 10 trials are performed and confidence intervals of 95% are used for every plot.

1) VN Downtime: Fig. 2 shows the VN downtime as a function of the VR memory size. According to the figure, we can observe that the VN downtime exhibits two distinct behaviours: either it does not depend on the VR memory or it does strongly depend on it. In the live migration and the clone migration without VR suspend, the VN downtime does not strongly depend on the VR memory size and it is almost constant for all the memory sizes evaluated.

The live migration has a VN downtime (or percentage of dropped packets) of 400 milliseconds (0.4%), and the clone migration has no downtime when using UDP traffic at 10Mbps. This behaviour is expected for both methods, where the downtime experienced on the VN live migration is mainly due to the XEN live migration procedure [7], which is an iterative process based on memory copy of dirty pages. In the VN clone approach, the VR memory RAM is copied at once, while the VR is still running, and at memory RAM write speeds.

In the case of the VN clone migration with VR suspend, the VN downtime does vary with the VR memory size and increases linearly with it. This is in fact due to the cloning phase of the VR, where the VR is put into suspend mode while its memory RAM is being copied. The VN clone migration without VR suspend outperforms the VN live migration approach and achieves no VN downtimes.

The traffic carried out by the VN does not significantly affect the $VN_{downtime}$, although a slightly higher percentage of dropped packets is registered for the UDP traffic with higher bitrate (i.e. 20Mbps).

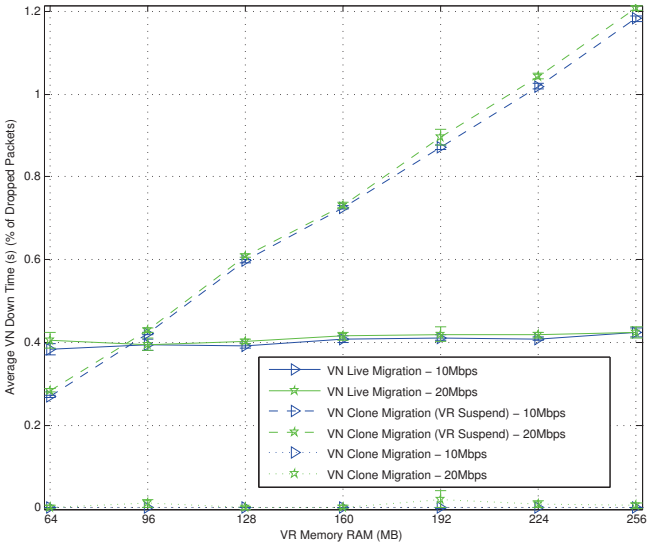


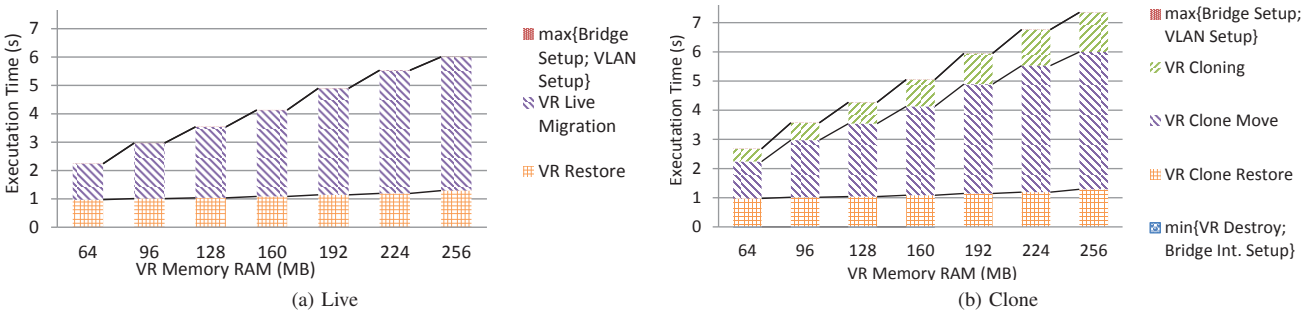
Figure 2. Virtual Network downtime (or Percentage of Dropped Packets) as a Function of the VR Memory RAM

2) VN Migration Execution Time: The VN migration execution time is illustrated in Fig. 3. On the left side (Fig. 3a), it is represented the execution time of the live migration process, and on the right side (Fig. 3b), it is represented the execution time of the clone migration process. The VN execution time grows linearly for both migration methods with the VR memory size. The live migration process is the one that takes less time to be fully performed, once it is performed internally by the XEN hypervisor.

In the clone migration process, the most time consuming operation is the VR Clone move (i.e. the relocation of the VR clone to the new physical host). This operation is influenced both by the VR memory size and by the available bandwidth at the physical path, between the physical host source and the physical host destination ($Mary$-$Susan$-$Bree$), which in our experiment was of 1Gbps (see formula 2). The second most time consuming operation is the VR cloning operation, which is also dependent on the VR memory size.

Both the bridging of virtual interfaces and the VLAN setup are the less time consuming operations, and take up to 4 and 8 milliseconds, respectively, to be performed. The total execution time of the clone migration process, with a VR of 64 MB memory RAM, is 2.75 seconds, while the total execution time of the live migration process is 2.36 seconds.

Notice that the total execution time of the clone migration process can be reduced, if it is also performed internally by the hypervisor. Moreover, this time can be further reduced if the VR clone relocation takes place in parallel with the VR cloning phase.



(a) Live (b) Clone

Figure 3. VN Migration Execution Time as Function of the VR Memory Size

V. Conclusion and Future Work

The VN migration process is an important mechanism both for network management purposes and for the conception and deployment of new network architectures and protocols based on network virtualization. The VN migration enables the virtual network to not be physically tied to a set of hosts previously assigned, and it makes it viable to move components of a VN or even the entire VN from one set of physical hosts to a new set on a seamless way and on-demand.

This paper proposed the VN clone migration as an alternative to the live migration approach. The VN clone migration performs cloning of the VR and transfers the VR clone to the new physical host. This approach requires no restrictions on the virtual network itself or on the networking protocols running inside of the virtual network. The results show that the proposed approach achieves no VN downtime, and it takes just a few seconds to be fully performed.

With respect to the situation where the VR is still running while the VR clone is being copied, we do not consider it as a drawback, if we take into consideration that we are assuming a VR as a $black-box$. We do agree that it is possible to perform a control plane update during the time slot where the VR is still running while the VR clone is being transferred. However, we assume that there exists re-transmission mechanisms on the networking protocols which are being operated inside of the VN (e.g. OSPF already provides this kind of mechanisms). Similar research works [6], [13] have addressed this issue by modifying the control of the VR; they are either not complaint with our assumption, or are dependent on the networking protocols which are configured on the VR. If we consider the scenario were VNs are being leased by Infrastructure Providers to Virtual network Operators, it would not be viable to change the control plane of the VR according to the networking protocols configured inside of the virtual network.

Future work spans from the integration and evaluation of the VR cloning approach with the network hypervisor, to the investigation and deployment of new VN (re-)mapping heuristics and even new mathematical formulations, using mixed integer optimization techniques, that will take into account the constraints imposed by the VN migration process (e.g. the maximum time that the embedding can take in critical situations - i.e. imminent fail-over or security attacks - or even the VN downtime).

Acknowledgment

The author was partially supported by the Portuguese Foundation for Science and Technology (FCT) with a scholarship No. SFRH/BDE/33751/2009. The research leading to these results has also received funding from the European Union Seventh Framework Programme (FP7) under grant agreement number 257448.

References

[1] L. Peterson, T. Anderson *et al.*, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 59–64, January 2003.

[2] T. Anderson, L. Peterson *et al.*, "Overcoming the Internet Impasse through Virtualization," *Computer*, vol. 38, pp. 34–41, April 2005.

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

[4] Y. Zhu, R. Zhang-Shen *et al.*, "Cabernet: connectivity architecture for better network services," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08. New York, NY, USA: ACM, 2008, pp. 64:1–64:6, ACM ID: 1544076.

[5] 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.

[6] 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.

[7] C. Clark, K. Fraser *et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 2005, p. 273–286.

[8] S. Lo, M. Ammar, and E. Zegura, "Design and analysis of schedules for virtual network migration," *Georgia Institute of Technology SCS Technical Report*, vol. GT-CS-12-05, July 2012.

[9] 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.

[10] 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.

[11] "Quagga routing suite," August 2012. [Online]. Available: http://www.nongnu.org/quagga/

[12] S. Avallone, S. Guadagno *et al.*, "D-ITG distributed internet traffic generator," in *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*, Sep. 2004, pp. 316 – 317.

[13] P. Pisa, N. Fernandes *et al.*, "Openflow and xen-based virtual network migration," *Communications: Wireless in Developing Countries and Networks of the Future*, p. 170–181, 2010.