OpenFlow and Xen-Based Virtual Network Migration*

Pedro S. Pisa, Natalia C. Fernandes, Hugo E. T. Carvalho, Marcelo D. D. Moreira, Miguel Elias M. Campista, Luís Henrique M. K. Costa, and Otto Carlos M. B. Duarte

Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ
Rio de Janeiro, Brazil

Abstract. Migration is an important feature for network virtualization because it allows the reallocation of virtual resources over the physical resources. In this paper, we investigate the characteristics of different migration models, according to their virtualization platforms. We show the main advantages and limitations of using the migration mechanisms provided by Xen and OpenFlow platforms. We also propose a migration model for Xen, using data and control plane separation, which outperforms the Xen standard migration. We developed two prototypes, using Xen and OpenFlow, and we performed evaluation experiments to measure the impact of the network migration on traffic forwarding.

1 Introduction

To experiment new alternatives in the Internet core using production traffic is considered unpractical. Internet providers do not feel comfortable to perform modifications that could damage their service. This problem causes a dead lock, on the one hand Internet must evolve to handle new demands, but on the other hand, new mechanisms that change the Internet cannot be applied in the underlying infrastructure. Many works tackle this problem by using virtualization techniques [4, 9] because, in a virtualized environment, the physical substrate is shared among many virtual networks. Virtual networks are isolated and, consequently, new proposals could be run together with the current Internet without disturbing the production traffic. Network virtualization requires new control and management primitives for redistributing physical resources among the virtual nodes. One of these primitives is the live virtual network migration. The idea is to move virtual networks among the available physical resources without disrupting the packet forwarding and network control services [11].

Virtual network migration allows dynamic planning of physical resources and traffic management on demand. Network migration can also be applied to create green networks [1], because virtual networks could be placed on different physical routers according to the traffic demand to reduce energy consumption. This concept is also compatible with the idea of cloud computing, which is

* This work was supported by FINEP, FUNTTEL, CNPq, CAPES, and FAPERJ.

an attempt to efficiently share power, storage devices, user applications, and network infrastructure over the Internet as if they were services [7]. Virtual network migration could also be used to solve security problems. For instance, a link shared by different virtual networks could be jammed because of a denial of service attack to one of the virtual networks. In this case, the non-attacked networks could be migrated to different physical routers until the attack is solved. In the current Internet, none of the above mentioned applications is possible because live migration services are not available.

In this paper, we compare and evaluate different migration models, according to the type of virtualization in use. First, we evaluate the use of Xen[1] standard migration to verify the impact of this migration model over a virtual network. We also analyze virtual network migration models based on plane separation. The plane separation is a technique for dividing each network node in two parts: a control plane, responsible for running all the control algorithms and for constructing the routing table; and the data plane, which is responsible for forwarding the traffic. We propose a model for applying the plane separation on Xen migration and we evaluate the advantages and the disadvantages of this new migration model. Also, we analyze the use of OpenFlow migration techniques, which is a platform natively based on plane separation for virtualizing switched networks. We present an algorithm for migrating virtual flows without impacting the traffic forwarding service. We also compare the OpenFlow link migration technique with the link migration in Xen.

Traditional management and control activities are also benefited by the use of network migration. The planned maintenance, in which a server or a router is rebooted or even turned off for a period of time for upgrading hardware and software, is a typical operation in any network. When a router service is restarted, the time for re-synchronizing the routing protocol states with the neighbor routers can lead to high delays. If we use virtual routers with live migration, the service downtime during planned management can be reduced to zero. Another service that takes advantage of the virtual network migration is the deployment of new network services. While a service is still under tests or is still an initiative with a low number of users, it can be placed on a smaller AS or in low capacity physical node. As the service develops, it can be moved to a more adequate infrastructure. With the current network model, the network administrator must invest in equipments for the new services without knowing the real significance of each service. Therefore, network virtualization with migration brings not only a service innovation incentive, but also an economic incentive in equipments, management, and power consumption.

We developed two prototypes for migrating virtual networks without losses in the data plane using Xen and OpenFlow. We evaluate these two prototypes according to the packet losses during migration and we compare them to Xen standard migration. Besides, we measured the delays for the data plane in Xen and OpenFlow prototypes. Hence, we can estimate the impact of the migration over the network and also the delay for migrating a virtual network. This delay

[1] Xen is a virtual machine monitor used for machine virtualization.

is important in situations where the virtual network migration is being executed because of high link usage. If migration takes too long, the link congestion will also persist for a long period causing messages losses.

The remainder of the paper is structured as follows. Section 2 presents an analysis of virtual network migration models according to the virtualization platform. It also presents the proposal for migration in Xen with plane separation and an efficient algorithm for migration in OpenFlow. In Section 3, we describe the developed prototypes and the experimental results. Finally, Section 4 concludes the paper.

2 Virtual Network Migration

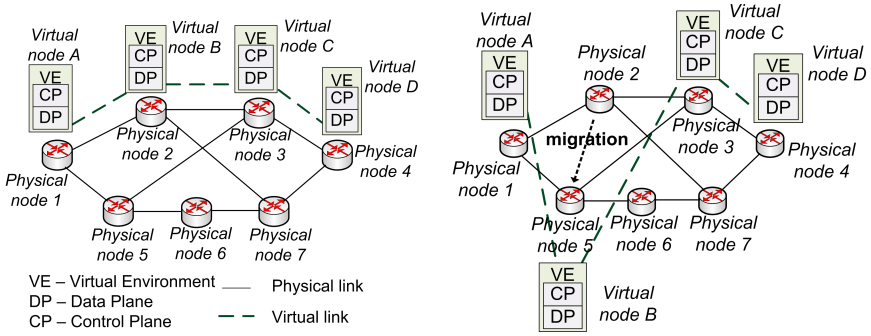
Virtual network migration is a function to rearrange the virtual network topology over the physical topology. Such rearrangement can even promote changes in the virtual network topology. The primary objective is to move virtual nodes and links minimizing the impacts on the virtual networks, such as requiring virtual routers to be reconfigured, disturbing virtual IP-level topology, or increasing the convergence time of the virtual network routing algorithms due to message losses caused by the migration process [12].

There are two basic approaches for virtualizing a network element [5]. Next, we show how the migration process works, according to the virtualization approach in use. We assume the existence of an entity that decide when and how to migrate nodes or paths in the network [2]. This entity can be implemented in a centralized way, for simplicity, or in a distributed way, for guaranteeing scalability, for instance. This entity is aware of the physical network topology and all the virtual network topologies and loads. As a consequence, this entity is able to define the new physical topology of a virtual network that will be migrated. Also, the existence of an arbiter raises an issue about security. Since we have an entity that has power over the whole network, the communication among this entity and the virtual/physical nodes must be completely safe. Moreover, the arbiter cannot be influenced by malicious network nodes that want to divert resources from one network to other.

2.1 1st approach: data and control planes on the same VM

In this virtualization approach, a virtual network is composed of virtual nodes, each one containing both data and control planes, and virtual links, as shown in Fig. 1(a). To maintain the virtual network topology, when migrating a node, we must find a new physical node that has the same virtual neighbors of the source node. Then, the whole virtual environment is migrated to the new physical node. For instance, in Fig. 1(b), the virtual node B is migrated from the physical node 2 to 5, because this modification does not change the virtual network topology.

One way of implementing this first virtualization approach is by using Xen, a virtualization platform created for commodity hardware [3]. In Xen architecture,



(a) Network view before the migration of virtual node B.

(b) Network view after the migration: no changes to the virtual topology.

Fig. 1. Virtual network migration assuming no data and control plane separation.

a virtual machine monitor (VMM), also known as hypervisor, is placed over the hardware and provides a virtual x86 hardware interface to virtual machines (VMs). Thus, each VM has its own virtual resources, such as CPU, memory, disk, and also its own operating system and application software. Xen also presents a privileged virtual machine, called Domain 0, which has full access to the physical devices and is responsible for providing reliable I/O-hardware support for the other virtual machines.

When we use Xen to create virtual networks, we assume that each VM works as a virtual router. Hence, to migrate a VM means to migrate a router. Because the VM is running a live service, we need to reduce the migration downtime, which is the time that the virtual machine is unavailable during the migration. It is also important to minimize the total migration time to guarantee that we can quickly free the resources of the initial physical machine. Xen has a built-in mechanism to migrate virtual networks [2]. This mechanism is based on some assumptions: the migration occurs inside a local network and the machine disk is shared over the network[2]. The main idea of this procedure is that migrating a virtual machine is the same of copying the memory of the virtual machine to the new physical location and reconfiguring the network links without breaking connections.

The simplest way to migrate the VM memory is to suspend the VM, transfer all the memory pages to the new physical location, and then resume the VM. To reduce the downtime, this procedure is evolved to a pre-copy migration, in which the memory copy is accomplished through two phases. The first phase,

[2] This shared disk assumption can be relaxed in our scenario, because routers from the same vendor usually implement the same small set of applications [11]. Then, we assume that the new physical router also has this set of programs and is able to load them onto the file system of the new VM. Hence, only virtual router memory and configuration files must be migrated.

called iterative pre-copy, transfers all memory pages to new physical machine, except for the 'hot pages', which are frequently modified pages. Consequently, the downtime is reduced, because only a few pages, instead of the whole memory, are transmitted while the VM is down. The next phase is called stop-and-copy. In this phase the VM is suspended and the hot-pages are transferred with the maximum transfer rate. Then, the new physical node confirms the reception of the whole memory to the old physical node.

The Xen built-in migration is inadequate for virtual networks due to the high packet loss rate during the VM downtime. Other problem of Xen built-in migration for virtual routers is that it assumes a migration within a local area network, which does not fulfill our objectives of migrating routers. Indeed, we cannot assume that physical nodes, such as nodes 1, 2, and 5 of Fig. 1, always belong to the same local network.

2.2 2nd approach: data and control plane separation

In the second virtualization approach, we separate data and control plane to migrate the control plane without message losses in data plane, which is an important characteristic for router virtualization. There are two different ways of implementing control and data plane separation. One way, which applies for virtualization platforms such as Xen, is run the control plane in the virtual environment, while the data plane runs in a shared area of the physical node. Another way is based on the OpenFlow platform [8], in which the network control is centralized in a special node, while each network node has a share data plane that is shared by different virtual networks

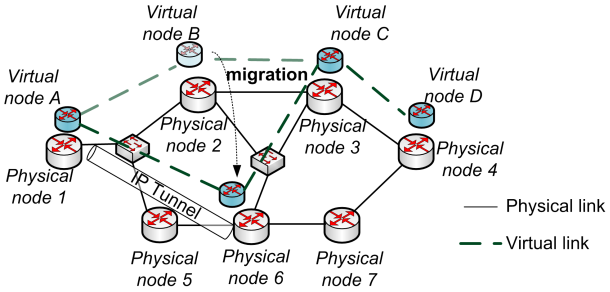
Migration in Xen with plane separation To reduce the packet loss in data plane during live migration, we propose to separate data plane and control plane in Xen. A similar approach was proposed for OpenVZ, a virtualization platform that provides multiple virtual user spaces over the same operating system [11]. Xen, however, presents a more programmable virtualization platform, because each virtual router can have its own software set, including operating system and protocol stack.

We developed a prototype that maintains in the VM the control plane, while the data plane is implemented in Domain 0. Each virtual router has its own forwarding table in Domain 0 and each table is a copy of the original forwarding table created by routing software running in the VM. When Domain 0 receives a control message, it checks which network the message belongs to and forwards the message to the corresponding VM. When Domain receives a data message, it is forwarded by Domain 0 using the forwarding table that corresponds to that virtual network.

The proposed migration mechanism works as follows. First, the Xen standard migration is started to migrate the VM. After the iterative pre-copy phase, the VM is paused and the remaining dirty memory pages are transferred. During this time, the data path is still working at Domain 0, with no interruptions or

packet losses. Also, a daemon is started in Domain 0 to buffer the control packets for the VM that is being migrated. When the whole memory is copied, the VM is resumed on the new physical machine (PM) and the network connections are created in the new Domain 0 using a dynamic interface binding module, which is responsible for mapping the virtual network interfaces to the physical network interfaces of the new PM. After that, a tunnel from the old PM to the new PM is created in order to transfer the control packets that were buffered in the old Domain 0 and also the new control packets. Finally, the ARP reply is broadcast to update the links and the data path in the old PM is removed.

Our migration mechanism guarantees no packet loss in the data plane during the VM migration, which is an important characteristic for a virtual router. Moreover, there is also no control packet loss. Our mechanism inserts only a delay in the control packet delivery. The proposed mechanism, however, is based on Xen default migration, which means that it still needs that routers are within the same local area network. In addition, the mapping of a virtual link over multiple physical links is still an open issue that depends on solutions such as IP tunnels or instantiating new virtual routers on the network. For instance, in Fig. 2, we migrate virtual node B from physical node 2 to physical node 6. Physical node 6, however, is not a one-hop neighbor of physical node 1. Consequently, to complete the link migration, we need to create a tunnel from physical node 6 to physical node 1 to simulate a one-hop neighborhood. The other solution is to instantiate a new virtual router to substitute the tunnel. This solution, however, modifies the virtual topology and influences the routing protocol functioning.



1) Migrate control plane from physical node 2 to 6 maintaining data and control planes.
2) Do interface dynamic binding and create tunnel between physical nodes 1 and 6.
3) Create a new forwarding table in Domain 0 of physical node 6
4) Reconfigure links with ARP reply
5) Delete data plane on physical node 2.

Fig. 2. Example of router migration in Xen when a virtual link is mapped to a multiple-hop path in the physical network.

Migration with OpenFlow OpenFlow is a platform to provide network virtualization which is natively based on plane separation [8]. In OpenFlow, all net-

work elements, such as routers, switches, and access points, work as "OpenFlow switches" and compose the data plane. Hence, OpenFlow networks are switched networks with support to virtualization. The control plane is centralized in a special element called "controller" [6], which runs in a separated machine. The controller knows the whole network topology and executes applications to configure flows according to the virtual network policies.

Network virtualization in OpenFlow is done in two different modes. In the first mode, there is one controller and each set of applications executing in the controller corresponds to each virtual network control plane. In the second mode, which provides more isolation between virtual networks, there is another machine besides the controller in the network, called FlowVisor [10]. In this mode, the control plane of each virtual network runs in a different controller and FlowVisor is responsible for giving/restricting the access of each controller to OpenFlow switches. In both modes, OpenFlow switches run a shared data plane, which is the concatenation of the forwarding tables of each virtual network. Indeed, the OpenFlow switches have no knowledge about the virtual networks or about which network each message being forwarded belongs to. OpenFlow switches are just hardware with a configurable forwarding table.

Since a controller is able to configure all OpenFlow switches, to migrate flows is the same of reconfiguring forwarding tables. The algorithm for migrating flows in OpenFlow is described in Fig. 3. First, when the controller decides to migrate the flows from a physical node to another, it creates new flow entries in each switch of the new path, except for the first common switch between the new and the old path, which is node 1 in the figure. After, the controller modifies the entries in this switch, redirecting the flows from the initial output port to the new output port. Finally, the controller deletes the old flow entries in the switches of the original path, which are node 2 and 3 in the figure.

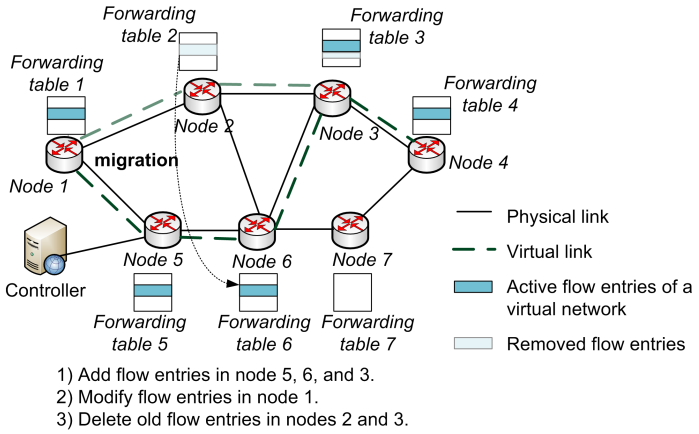


Fig. 3. Example of flow migration in OpenFlow.

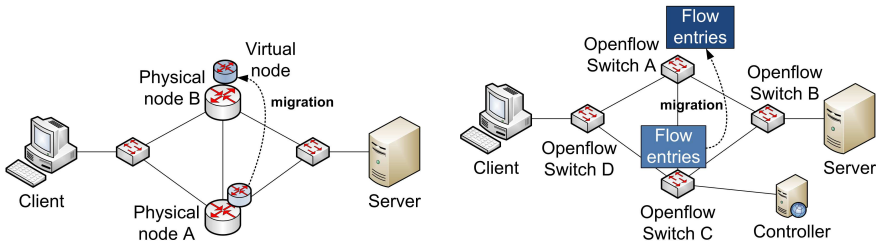
The advantages of OpenFlow migration are that it avoids the need for a local network, such as in Xen, because OpenFlow assumes a wide area switched network with configurable forwarding tables. Moreover, since we have a switched network and a centralized controller, there is no need to create tunnels after a migration. Since the control plane is now centralized and has a whole view of the physical network, there is no need to maintain the virtual topology. In Xen, it is necessary to guarantee that the communication between the shares of the control plane in each physical node is not impacted by the migration. Indeed, when we have a distributed control plane, a modification in the network topology will cause an update in the routing data and, consequently, a convergence delay until all nodes agree on the same routes again. OpenFlow provides an easier infrastructure for reallocating network resources. It is, however, based on a centralized controller, which may not scale for large area networks.

3 Developed Prototypes and Experiments

We implemented a prototype built on Xen with plane separation and a prototype built on OpenFlow. In Xen prototype, we modified the Xen standard packet forwarding scheme, because Xen architecture confines both data and control planes inside a virtual machine. Our prototype implements the plane separation with a mechanism that synchronizes the data plane, placed at Domain 0, with the routing information base (RIB) created by the control plane inside the virtual machine. The implemented migration mechanism uses the Xen standard migration mechanism to migrate the control plane as its first step. After that, the destination Domain 0 creates a forwarding table to the migrating virtual router and synchronizes the forwarding table with the RIB on the virtual machine. Finally, the link migration is done through ARP replies and then the source Domain 0 removes the old data plane. In OpenFlow prototype, we migrate the virtual data planes by moving flows from the old to the new physical switches. We developed a NOX application that creates a flow from "Client" to "Server" and vice-versa when the network is started, as shown in Fig. 4(b). When the migration is started, the controller constructs new flow table entries on the new path and, after that, delete the old flows entries, as described in Section 2.2. Accordingly, there is no packet loss during OpenFlow migration.

We developed a testbed, described in Fig. 4, to evaluate the presented migration prototypes, comparing with the Xen standard migration, which has no plane separation. In the Xen scenario, we have two physical machines, Physical Node A and Physical Node B running Xen 4.0. The experiment consist of generating an UDP data traffic and migrating a virtual router from Physical Node A to Physical Node B while the virtual router forwards the data traffic from the client to the server. To not disturb the data traffic, we use a separated link to transmit the migration traffic during the migration process. In the Xen standard migration, the data traffic is forwarded by the virtual machine. In the Xen with plane separation, the data traffic is forwarded by the shared data plane in Domain 0 and the control traffic is forwarded by the virtual machine.

In the OpenFlow scenario, the two physical machines with Xen are replaced by OpenFlow switches, running OpenFlow v 0.8.9. There is also a centralized NOX controller that accesses and controls all OpenFlow switches.



(a) Xen scenario. (b) OpenFlow scenario.

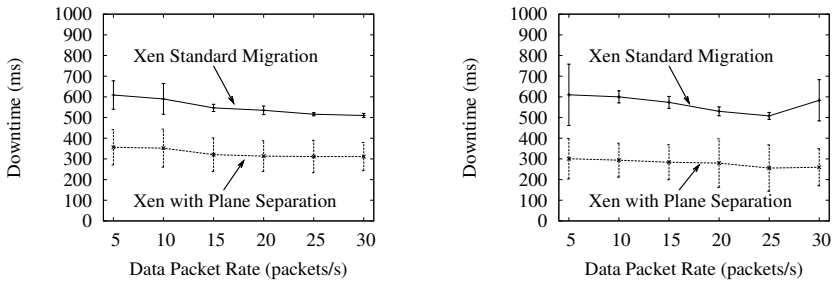
Fig. 4. Experimental scenarios for virtual network migration.

Results

In our experiments, we measured the delays and losses due to the migration with different data packet rates. We present experiments with 64-byte and 1500-byte packets, which respectively represent the minimum Ethernet payload and the most common Ethernet MTU, but there were no significant differences in the results when the packet size varies.

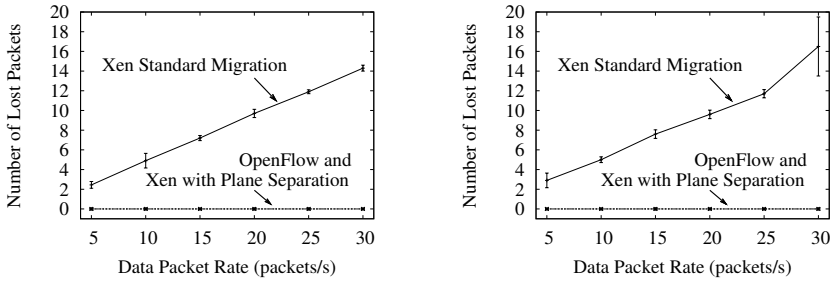
Fig. 5 presents the time elapsed during the downtime stage. The results show that the downtime is roughly constant with the growth of the data packet rate for all migration schemes. Compared with the Xen standard migration, the Xen with plane separation has a downtime that is 200 milliseconds lower on the average. This difference exists because in Xen standard migration, the virtual machine must forward all data and control packets, which increases the number of 'hot pages' in virtual machine memory and, consequently, raises the downtime. In the Xen with plane separation, the data traffic is forwarded by Domain 0 and the memory pages dirtied in the virtual machine come only from the routing software running in the virtual machine, which reduces the downtime.

Fig. 6 shows the number of data packets that were lost during downtime in the experiments. As expected, in the Xen standard migration, the number of lost packets increases linearly with the packet rate, because the downtime is constant and the data packet rate grows linearly. OpenFlow and Xen with plane separation tackle this issue. Xen with plane separation has a downtime only during the control plane migration and the data plane is not frozen during migration. In OpenFlow, the migration mechanism moves the data traffic to a new path without migrating the control plane. Indeed, OpenFlow has no downtime, since neither the data plane nor the control plane are stopped due to the migration.



(a) Packet size: 64 bytes. (b) Packet size: 1500 bytes.

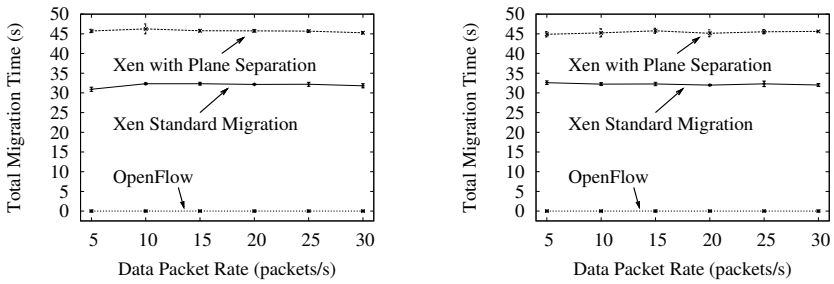
Fig. 5. Migration downtime as a function of the data packet rate.



(a) Packet size: 64 bytes. (b) Packet size: 1500 bytes.

Fig. 6. Number of lost packets during downtime as a function of the data packet rate.

Fig. 7 shows the total migration time, which consists of the time between the migration triggering and the moment when the data packets start to pass through the destination machine. The difference between Xen standard mechanism and Xen with plane separation is about 15 seconds. This variation occurs in our prototype due to data plane synchronization after control plane migration and the remapping of the virtual interfaces to the appropriate physical interfaces, which does not exist in the Xen standard migration. OpenFlow total migration time is about 5 milliseconds, because OpenFlow only migrates the data plane. This time comprises the interval between the beginning and the ending of sending flow change messages from the controller to all switches. In our scenario, the controller reaches all the switches with two hops at the most, as we can see in Fig. 4(b). Nevertheless, if the distance between the controller and the OpenFlow switches rises, the total migration time increases. Besides, if the number of migrated flows grows, the total migration time also increases. This happens due to the increased number of messages sent to the switches.



(a) Packet size: 64 bytes. (b) Packet size: 1500 bytes.

Fig. 7. Total migration time as a function of the data packet rate.

4 Conclusions and Future Directions

We evaluated the impact of different virtual network migration models with Xen and OpenFlow virtualization platforms. We observed that data and control plane separation is a key feature for reducing packet losses during the migration. Moreover, in the proposed migration mechanism for Xen, the packet forwarding through Domain 0 also reduced the number of dirtied pages in the virtual machine. Therefore, the downtime of the control plane during migration is reduced by 200 milliseconds. We obtained a significant difference in the packet loss rate and control plane downtime during migration when comparing the Xen standard migration and the proposed migration mechanism with plane separation for Xen. Besides, the analysis of the proposed migration algorithm for OpenFlow showed that it is an efficient approach for migrating virtual networks, because it provides a downtime of less than 5 milliseconds and no packet losses during the migration process.

The control plane downtime and the mapping of a virtual link over multiple physical links are the main observed drawbacks in the Xen migration. OpenFlow has none of these disadvantages, but it is based on a centralized controller, which can restrict the size of the network. Other approaches for migrating the data plane in Xen should also be analyzed, such as the instantiation of new virtual routers instead of mapping virtual links on multiple physical links. In addition, we intend to analyze the real impact on the traffic forwarding of buffering the control plane messages during downtime in the Xen migration model assuming data and control plane separation.

Bibliography

[1] Bolla, R., Bruschi, R., Davoli, F., Ranieri, A.: Energy-aware performance optimization for next-generation green network equipment. In: PRESTO'09: Proceedings of the 2nd ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow. pp. 49–54 (2009)

[2] Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: NSDI'05: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation. pp. 273–286. USENIX Association, Berkeley, CA, USA (2005)

[3] Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Mathy, L., Schooley, T.: Evaluating Xen for router virtualization. In: ICCCN'07: International Conference on Computer Communications and Networks. pp. 1256–1261 (Aug 2007)

[4] Feamster, N., Gao, L., Rexford, J.: How to lease the Internet in your spare time. ACM SIGCOMM Computer Communication Review 37(1), 61–64 (Jan 2007)

[5] Fernandes, N.C., Moreira, M.D.D., Moraes, I.M., Ferraz, L.H.G., Couto, R.S., Carvalho, H.E.T., Campista, M.E.M., Costa, L.H.M.K., Duarte, O.C.M.B.: Virtual networks: Isolation, performance, and trends. Tech. rep., Grupo de Teleinformática e Automação - GTA/COPPE/UFRJ, Rio de Janeiro, Brazil (Mar 2010)

[6] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: NOX: Towards an operating system for networks. ACM SIGCOMM Computer Communication Review 38(3), 105–110 (Jul 2008)

[7] Han, S.M., Hassan, M.M., Yoon, C.W., Huh, E.N.: Efficient service recommendation system for cloud computing market. In: ICIS'09: Proceedings of the 2nd International Conference on Interaction Sciences. pp. 839–845 (2009)

[8] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., S., Turner, J.: OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38(2), 69–74 (Apr 2008)

[9] Ratnasamy, S., Shenker, S., McCanne, S.: Towards an evolvable Internet architecture. ACM SIGCOMM Computer Communication Review 35(4), 313–324 (2005)

[10] Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., Huang, T.Y., Kazemian, P., Kobayashi, M., Naous, J., Seetharaman, S., Underhill, D., Yabe, T., Yap, K.K., Yiakoumis, Y., Zeng, H., Appenzeller, G., Johari, R., McKeown, N., Parulkar, G.: Carving research slices out of your production networks with OpenFlow. ACM SIGCOMM Computer Communication Review 40(1), 129–130 (2010)

[11] Wang, Y., Keller, E., Biskeborn, B., der Merwe, J.V., Rexford, J.: Virtual routers on the move: Live router migration as a network-management primitive. In: ACM SIGCOMM. pp. 231–242 (Aug 2008)

[12] Wang, Y., der Merwe, J.V., Rexford, J.: VROOM: Virtual routers on the move. In: Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking. pp. 1–7 (Nov 2007)