CLOUD SERVICES, NETWORKING AND MANAGEMENT

CLOUD SERVICES, NETWORKING AND MANAGEMENT

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Library of Congress Cataloging-in-Publication Data:

Title, etc
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

CONTENTS IN BRIEF

CONTENTS

CHAPTER 1

VIRTUAL MACHINE MIGRATION

Diogo M. F. Mattos, Lyno Henrique G. Ferraz, and
Otto Carlos M. B. Duarte.

Grupo de Teleinformática e Automação (GTA/UFRJ)
PEE/COPPE - DEL/Poli
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil

Virtual Machine Migration is one of the most important benefits of virtualization because it provides facilities such as user mobility, load balancing, energy savings, system up-dates, maintenance, and failure management. Different virtual machine migration procedures exist concerning online or offline migration, and live migration. Virtual network live migration without packet losses is particularly interesting because I/O procedures are not yet fully virtualized by current hypervisors. Seamless virtual machine migration must overcome different level of complexity when source and destination physical machines belong to the same cluster, or to the same local area network or to different clouds interconnected by a Wide Area Network.

This chapter details the pros and cons of each virtual machine migration model and describes different proposal for guaranteeing an efficient migration procedure.

1.1 Introduction

Cloud Computing is experiencing an extraordinary growth [1, 2, 3, 4, 5]. Besides, virtualization technologies are widely adopted by companies to manage flexible computing environments and to run isolated virtual environments for each customer [6]. Virtualization also provides the means to accomplish efficient allocation of resources and to improve management, reducing operating costs, improving application performance and increasing reliability. Virtualization logically slices physical resources into virtual environments, which have the illusion of accessing the entire available physical resource. Hence, the physical machine resources are shared between multiple virtual machines (VMs), which run their own isolated environment with an operating system and applications. By decoupling virtual machines from their underlying physical realization, virtualization allows flexible allocation of virtual machines over physical resources. To this end, virtualization introduces a new management primitive: virtual machine migration [7]. Virtual Machine migration is the relocation of virtual machines over the underlying physical machines, even if the virtual machine is still running.

The VM migration primitive enhances user mobility, load balancing, fault management, and system management [8]. The migration that occurs without the interruption of services running is called live migration.

Virtual machine migration is similar to the process migration, but it migrates a complete operating system and its applications. Process migration moves a running process from one machine to another. Process migration is very difficult, or even impossible to accomplish, because processes are strongly bound to operating systems, by means of open sockets, pointers, file descriptors and other resources [8]. Unlike process migration, VM migration moves the entire operating system along with all the running processes. Migrating an entire operating system with its applications is a more manageable procedure, and is facilitated by the hypervisor, which exposes an interface between physical machine and the VM operating system. The details of what is happening inside the VM can be ignored during migration. The VM migration also has challenges inherent security to transfer the state of a virtual machine across physical machines and to establish a trustworthy computing environment on the destination physical machine.

In the context of virtualization, it is necessary to ensure that virtual environments are secure and trustworthy. Thus, the hypervisor, which is a software layer responsible for creating the hardware abstraction to the virtual environment, must implement a trusted computing base (TCB) [9, 10]. Indeed, the TCB is divided into two parts: the hypervisor and an administrative domain, as show in Figure 1.1. The hypervisor controls the hardware directly and executes at the highest privilege level of the processor. The administrative domain is a privileged VM that controls and monitors other VMs. The administrative domain have privileges to start and stop VMs, to run guest VM configuration, to use and to monitor physical resources, and to run I/O operation directly on the physical devices for the virtualized domains. This common architecture for virtualized systems creates, however, security challenges, such as lack of privacy of guest VMs. Administrative domain runs in a privileged level to inspect the state of guest VMs, such as the contents of its registers into memory and vCPUs. This privilege can be usurped by attacks on the software stack in the administrative domain and by malicious system administrators [11]. Therefore, it is necessary to establish a Trusted Computing Base (TCB) on the hypervisor and on the administrative domain to ensure the security of virtualized environments.

Specific relevance is given to a hybrid virtualization system based on Xen and OpenFlow platforms, called XenFlow [12], which focuses on router virtualization, especially on the virtual router migration without packet losses. XenFlow provides migration of virtual topologies over the physical realization, performing both migration of virtual routers to another physical host and remapping virtual links on one or more physical links. This feature allows to extent virtual-router migration when compared to the other proposals in literature [13, 14, 15], because routes are remapped to any destination physical node by means of OpenFlow network.

This chapter presents the major VM migration techniques. This work highlights the benefits, costs and challenges for the realization of the live migration of virtual machines.

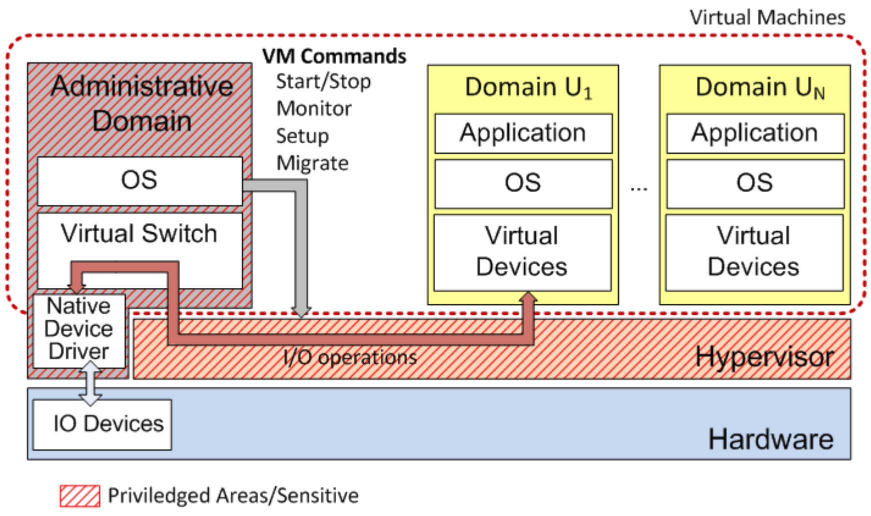


Figure 1.1 General Xen-based virtualization architecture. The hachured areas, Administrative Domain and Hypervisor, indicate the most sensitive software modules because they run on highest privilege level.

We highlight I/O virtualization techniques and discuss how to migrate virtual machine even if they directly access I/O devices or use I/O virtualization techniques. The chapter sets out the main security requirements to be ensured during the migration of virtual environments. Then, we examine various schemes of VM migration and discuss research directions in virtualization security. The ultimate goal is to provide a deep understanding of the developments and the future directions regarding virtualized environments migration primitive.

The rest of this chapter is organized as follows. Section 1.2 sets a background for understanding virtual machine migration and its challenges. Virtual network migration is explained on Section 1.3, in which we also present a proposal for migrating virtual routers without packet losses. The main security requirements and proposals for virtualized environments are identified on Section 1.4. Future research directions and open challenges are discussed in Section 1.5. Section 1.6 concludes this chapter.

1.2 Virtual Machine Migration

The procedure of migrating the operating system and applications from a physical machine to another physical machine is an important feature in a virtualized environment. Virtual Machine (VM) migrations encompass four main resource transferring: processor, memory, network, and storage [15]. During the migration process, the VM is paused on source host and is resumed on the destination host only when all resources have already been migrated and configured into the new host. The VM stays offline during a period of time, called downtime, which corresponds to time when the VM is paused until its resumption at the destination. The downtime period varies according to the resources available to the VM, to the workload submitted to VM, and to the migration technique: offline or live migration.

1.2.1 Offline and Live Migration

Offline Migration transfers the VM to destination physical host while the VM is off. The offline migration introduces a great delay in services of VM but it is the easiest to accomplish because it does not require the VM state preservation. As the VM is off, there is no network connections to preserve and it is neither necessary to transfer the processor state nor the RAM content. The offline migration procedure just comprises shutting down and restarting the VM into another location.

The storage migration, or disk migration, is accomplished by standard data transfer tools and is the only network traffic generated. It takes a long time and a lot of network bandwidth to transfer a whole disk. As a matter of fact, VM migration is usually accomplished within a LAN with a Network-Attached Storage (NAS) device that allows a VM to access its disk from anywhere in the network, which makes unnecessary to migrate the disk.

Live migration transfers the VM while it still runs. The live migration should not cause a perceptible downtime to the VM user. Assuming the source and destination physical machines in the same LAN with a NAS, live migration only should transfer the state of the processor, the state of the memory and network connections.

The processor live migration consists of creating a virtual CPU (vCPU) for the virtual machine at the destination and copying the vCPU state from source to destination physical machines. Nevertheless, this task becomes complex when the source and the destination host processors are different. Migrations between different processors of the same manufacturer require the same instruction sets to work properly. In these cases, as a consequence, it is necessary to limit the instruction set of the virtual CPU to a common instructions set of both processors. This operation is called CPU mask.

The network live migration procedure should maintain the Internet Protocol (IP) address of the source VM to preserve all open transmission control protocol (TCP) connections. To keep the same IP address at the destination, it is very simple when the source and destination physical machines are in the same Local Area Network (LAN). In this case, the destination physical machine generates gratuitous Address Resolution Protocol (ARP) replies to advertise the new physical location of the virtual machine, that is, only the advertisement of the Medium Access Control (MAC) address of the migrated VM is required. Otherwise, when the source and destination machines are not in the same LAN, network redirection mechanisms would be required due to the localization semantic of the IP address.

Live memory migration is the transfer of memory contents from the source to the destination host taking into account memory changes during the migration procedure, called retransmission of dirty pages. Venkat [16] divided the memory migration into three phases:

- Push phase: While the source VM is running, its memory pages are transferred to the destination VM. If a page is modified after being transferred, it is necessary to resend this page to avoid failures.

- Stop-and-copy phase: As the name suggests, the source VM is stopped, then the memory pages are transferred.

- Pull phase: The destination VM is started and generates a page fault when it tries to access a page that was not copied yet. This fault requests the page to be transferred from the source to the destination.

Two live migration strategies [16]: pre-copy and post-copy, only use a combination of two of the above-mentioned phases.

The pre-copy live migration strategy applies the phases: push and stop-and-copy. First, an empty VM is created at the destination physical host and the migrating VM memory pages are copied to the VM at destination physical machine, while the VM still runs on the source host. During this process, the running VM rewrite the memory pages which are re-sent to destination host. This push phase ends when one of the two conditions are reached: (i) The number of dirty pages per iteration are small enough to cause a short downtime period; (ii) The push phase reaches a maximum number of iterations. After the push phase, it comes the stop-and-copy phase, in which the VM is suspended at the source host, the remaining dirty pages are transferred to the destination host, and the VM is resumed on the destination host. The downtime varies according the workload from tens or hundreds of milliseconds to a few seconds [15]. It is important to notice that determining when to stop the push phase and start the stop-and-copy phase is not trivial. Stopping the push phase too soon can result in longer downtime, as more data will be transferred after suspending the VM. On the other hand, stopping too late results in longer total migration time and network bandwidth occupation, as more time will be spent re-sending dirty pages. Therefore, there is a trade-off between total migration time and downtime. The pre-copy procedure requires the verification of memory pages to send them to the destination through the network. These CPU and bandwidth consumption should be monitored to minimize service degradation. Xen uses pre-copy as its live migration strategy [14].

The post-copy live migration strategy use the phase stop-and-copy first, then the pull phase. First, the VM is suspended at the source and few VM execution states are trans-

Table 1.1 Comparison of Offline and Live migration techniques.

Characteristic / **Technique**	**Storage Migration**	**Memory Migration**	**Network Migration**	**Downtime**	**Total Migration Time**
Offline migration. Shutdown VM and restart at destination host.	Standard copying tools, if migrated.	Not transfered. Loss of volatile data.	Reconfiguration at destination host. Network connections not migrated.	Long period of time. VM and services restart (if storage is not migrated).	Equal to downtime (if storage is not migrated).
Live migration. Transfer running VM to destination host.	Not migrated. Storage is accessable through the network.	Transfered. Pre- and post-copy and retransmission of updates.	Reconfiguration at destination host. Transfer of network state. Network connections preserved.	Short period of time. VM pause/resume.	Equal to downtime plus memory transfer time. Vary according to the workload that requires retransmission of dirty pages.

ferred to the destination host, namely CPU registers and non-paged memory. The VM is resumed at the destination despite the absence of many memory pages, which still are at the source host. The source host begins to send the remaining memory pages. The destination host generates faulty memory accesses when the VM tries to access memory pages that were not transferred yet. These faulty memory accesses are sent back to the source host, which prioritizes the requested memory pages to send. This process can degrade memory intensive application performance, but cause minimal downtime. There are some ways of handling page fetching in order to increase performance, such as:

- Active Pushing: the pages are pro-actively pushed from the source to the destination. Page faults are handled with priority over non-critical pages.

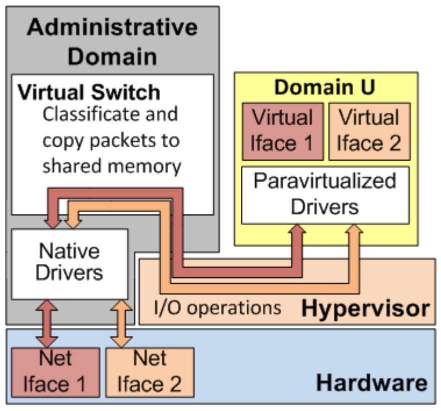
- Pre-paging: an estimation of memory access pattern is generated to allow the active pushing of the pages that are most likely to generate faults.

Table 1.1 compares Offline and Live migrations.

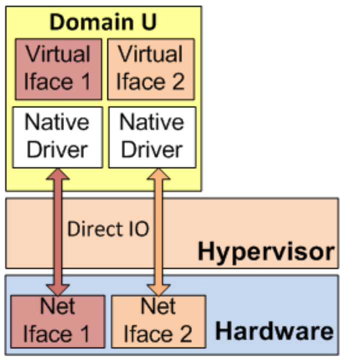
1.2.2 I/O Virtualization and Migration of Pass-through Devices

Input/Output (I/O) virtualization of network devices is challenging because current network interface controllers are unable to distinguish which specific virtual machine is writing to or reading from the shared memory space. Therefore, a controller or a hypervisor must redirect (multiplexing or demultiplexing) data to/from specific memory area in an administrative domain from/to different virtual machine shared memory areas. This procedure negatively impacts the performance, since it introduces extra memory copies, it centralizes the interruption handling at administrative domain processing time slice, and it demands execution of software instructions for multiplexing data in administrative domain, such as virtual bridges, as shown in Figure 1.2(a). Thus, a technique to improve I/O device performance is the use of pass-through technologies to avoid the centralization and memory copies by providing direct I/O procedures to/from the virtual domain from/to the physical device. Although the pass-through technology improves I/O virtualization performance, the pass-through device belongs to a single VM and cannot be shared by other VMs, as shown in Figure 1.2(b).

The main technique to provide direct I/O virtualization is Single Root I/O Virtualization (SR-IOV) for Peripheral Component Interconnect Express (PCIe) [17]. The specification SR-IOV stands for how PCIe devices can share a single root I/O device with multiple virtual machines. Indeed, a SR-IOV enabled hardware provides several PCIe virtual functions to the hypervisor, which can be assigned directly to virtual machines as pass-through devices, as shown in Figure 1.3(a). Besides SR-IOV, Intel also proposes Virtual Machine Device Queues (VMDq) [4] for network I/O virtualization. VMDq technology enabled network device has separated queues for virtual machines. The network interface classifies received packets to the queue of a VM and fairly sends packets of all queues in round robin manner. As VMDq applies a paravirtualized device driver, it uses shared pages to avoid packet copying between the virtual network interface in the VM and the physical network queue. The VM benefits from faster classification and a paravirtualized device driver, while SR-IOV technology exposes a unique device interface to the virtual machine. The implementation of VMDq paravirtualized driver assures better performance than paravirtualized network drivers. Besides, VMDq paravirtualized driver support live migration in a similar way than when using common paravirtualized drivers [4], illustrated by Figure 1.3(b).

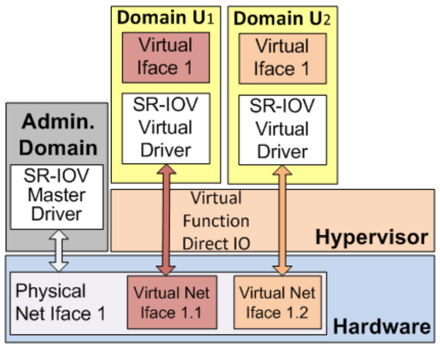


(a) Network I/O virtualization with paravirtualized drivers. Administrative domain centralizes all I/O operations.

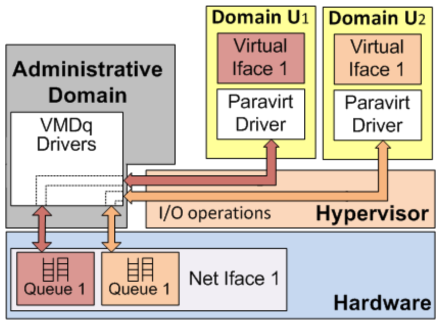


(b) Direct I/O network virtualization. A network interface card is directly connected to virtual machine.

Figure 1.2 I/O virtualization modes.



(a) Network I/O virtualization with SR-IOV. Virtual machines directly access NIC virtual functions.



(b) Network I/O virtualization with VMDq. Virtual machines access device queues through a paravirtualized driver.

Figure 1.3 Hardware-assisted network I/O virtualization modes.

VMWare and Intel propose Network Plug-In Architecture (NPA/NPIA) [4] to live migrate pass-through devices. The proposal creates a new driver for virtual machine, which allows the online switching between SR-IOV and paravirtualized devices. This technology designs two new software modules: a kernel shell and a plug-in for the virtual machine. Kernel shell acts as an intermediate layer to manage pass-through devices, which implements a device driver for the SR-IOV device. Plug-in, in its turns, implements virtual functions of the device, as a device driver, but interfaces with kernel shell instead of directly controlling the device, exposing a virtualized network interface card to the virtual domain. The kernel shell provides a hardware abstraction layer and the plug-in implements hardware communication through the kernel shell. Plug-in may be plugged or unplugged on the fly. To reduce migration downtime, while performing plugging/unplugging actions, the hypervisor employs an emulated network interface. This technology trivially supports live migration because a virtual network interface can be unplugged while running the VM. On the other hand, a drawback of the this approach is the need for rewriting all the network device drivers, which may limit its adoption [4].

Pass-through I/O virtualization technology improves virtualized device performance by making a tight coupling between the VM and the hardware device. Thus, virtual machine live migration becomes more difficult because pass-through devices are totally controlled by virtual machine and the hypervisor does not access the internal states of the device. Indeed, in pass-through I/O virtualization the hypervisor does not interfere into the communication between the physical device and the virtual machine. Therefore, the internal states of the physical device must be migrated with virtual machine, in order to accomplish a successful live VM migration [4].

A way to migrate VM with pass-through devices is to let user stop everything using a pass-through device, and then migrate and restore the virtual machine into the destination physical host. Although this method works, it is not generic enough to fit all operating systems, it involves a greater downtime, it needs to be inside the virtual machine, and it needs a lot of intervention of the user [18]. A generic solution to suspend the VM before migrating is Advanced Configuration and Power Interface (ACPI)[1] S3 [18]. Sleep state S3 stands for the sleep or suspend state of a machine, in which the operating system freezes all process, suspends all I/O devices, and then goes to the sleep state but the RAM remains powered. It is worth noting that in sleep state all context is lost, except for the system volatile memory. The major drawback of this approach is that whole system is affected, inducing a long service downtime besides disabling the target device.

Migration of a pass-through I/O device may also be accomplished by the PCI hotplug mechanism [18]. Migrating a VM using PCI hotplug work as follows. Before live migrating, in the source host, the entity responsible for the migration triggers an event of hot unplugging the virtual PCI pass-through device against the guest VM. Then, the migrating VM responds to the hot unplugging event, and stops using the device after unloading its driver. Without running any pass-through device, the VM can be safely live migrated to the destination host. After the live migration, in the destination host, it triggers an event of hot plugging a virtual PCI pass-through device against the VM. Eventually, the guest VM loads the appropriate driver and starts using the new pass-through device. As the guest reinitializes a new device, that has nothing to do with the old one, it should reconfigure it as the previous one.

CompSC proposes a live migration mechanism for VM using pass-through I/O virtualization [4]. The key idea of CompSC is to change as less as possible the code of drivers and prevent the hypervisor to have any specific knowledge about the migrating device. The hypervisor examines the list of registers of the network device and saves them into the shared memory area. The hypervisor does not know the list of registers *a priori*. For this reason, the hypervisor gets this list of registers also from the shared memory area, where the device driver places it during the boot process. The device driver completes the state transferring between hosts. Every time before the driver releases a read lock, it stores enough information about the latest operations or set of operations to achieve a successful resume. In the resume procedure, the device triggers the target hardware using the same saved state information. The proposal also provides a layer of self-emulation, which can be placed in the hypervisor or in the device driver. Placing the self-emulation layer in hy-

[1]Advanced Configuration and Power Interface (ACPI) specification is an open standard for device configuration and power management by the operating system. This standard replaces some other standards bringing power management under the control of the operating system instead of BIOS control as stated by the replaced standards.

pervisor, the hypervisor intercepts all accesses to emulated registers and returns the correct

Table 1.2 Comparison of migrating I/O virtualization techniques.

Characteristic / **Technique**	**Pros**	**Cons**	**Sumary**
SR-IOV	Good performance. VM direct access to device.	Hard to migrate. Hypervisor cannot access device state.	Hardware provides multiple virtual functions. Hypervisor sets virtual functions to VMs. VM interacts directly with hardware devices.
VMDq	Good performance and easy to migrate. Packet classification by hardware and conventional paravirtualized driver.	Slightly performance degradation. Minor driver domain participation in I/O.	VMDq driver writes and reads packets directly on shared pages in driver domain which avoid packet classification and extra packet copies.
NPA-NPIA	Good performance and easy to migrate. Hotplug of SR-IOV virtual functions and paravirtualized drivers.	Hard to deploy. New virtual network device drivers in VMs.	It creates a pair "Kernel Shell and Plug-in", which allows Plug-in to be migrated carrying all device states, while Kernel Shell implements virtual function into the driver.
Pause/Resume	Easy to deploy. It uses current technologies.	Hard to migrate and loss of volatile data. It depends on users' interaction. VM suspension.	VM is suspent on source host and, after, it is resumed on destination host.
PCI Hotplug	Good performance and easy to deploy. It uses current technologies.	Loss of volatile data. Pass-through devices hotplugging.	Source host unplugs the virtual PCI pass-through device of VM. After migration, a new pass-through device is loaded and reconfigured on the migrated VM.
CompSC	Good performance and easy to migrate. VM uses current technologies. Easy to deploy. Hypervisor uses new live migration software.	Slightly performance degradation during migration. It uses emulated virtual network device driver.	Hypervisor saves pass-through device states before migration, and restores the device state after migration.

value. A layer of self-emulation in the driver processes the fetched value and corrects it after the access. A layer of self-emulation in hypervisor requires only the list of emulated registers and requires few code changes to the driver, but the performance degrades due to interception of I/O operations. A layer of self-emulation in device driver requires less overhead, but produces more code changes [4]. Table 1.2 summarizes the migration proposals of main pass-through I/O virtualization techniques.

1.3 Virtual Network Migration without Packet Loss

Network virtualization is the technique that decouples network functions from their physical substrate, enabling virtual networks to run logically separated and over the a physical network topology [19]. The logical separation enables virtual network migration, which allows online physical topology changes avoiding reconfiguration, traffic disruption and long convergence delays [13]. The virtual network migration consists of migrating the virtual network element, also called virtual router, to another physical location, without packet losses or losing connectivity. The key idea to avoid packet losses is the separation of control and data planes, the former responsible for performing control operations, such as running routing protocols and defining QoS parameters, and the latter responsible for the packet forwarding [13, 14]. As the virtual router should always forward the traffic, the data plane is copied to the physical host while the virtual router migrates. After the migration, the data plane in source host is deactivated, so the virtual router completely runs in the new location.

Both Wang *et al.* and Pisa *et al.* use plane separation paradigm to migrate virtual routers without packet losses [13, 14]. They assume an external mechanism for link migrations to preserve neighborhood after migration, such as maintaining the same set of neighbors or tunneling. Pisa *et al.* assume all physical routers connect to the same Local Area Network (LAN) to facilitate link migration [14]. On the other hand, flow migration on the OpenFlow platform is easy. Pisa *et al* present an algorithm that is based on the redefinition of a flow path in the OpenFlow network [14]. This proposal has zero packet losses and low overhead of network control messages. Although, this migration proposal is limited to OpenFlow switched networks, and it is not applicable to router virtualization systems.

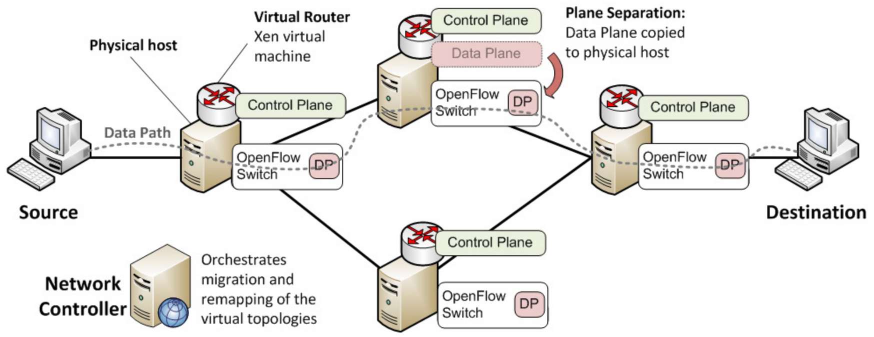


Figure 1.4 XenFlow architecture overview. Xen virtual router data plane is copied to physical host OpenFlow switch. Network controller orchestrates virtual router and link migration.

Mattos and Duarte present XenFlow [12], a hybrid network virtualization system based on plane separation paradigm with Xen and OpenFlow platforms [20, 21] to both migrate virtual routers and virtual links. Virtual machines act as the routers control plane running routing protocols, and data planes of all virtual routers run centrally in the Xen administrative domain Domain 0. Physical machines have an OpenFlow switch to connect Xen virtual machines to the physical network, and each Xen virtual machine acts as generator of rules to these switches. The remapping of the virtual topologies is orchestrated by a network controller capable of acting on the OpenFlow switches and of triggering the migration of virtual machines on any network node. Figure 1.4 presents this architecture. The architecture allows to migrate virtual routers beyond a local area network, because routes are remapped to any destination physical node by means of OpenFlow network. However, the architecture forces all virtual networks to share the same data plane, violating the requirement of isolation between virtual environments. Thus, XenFlow isolates virtual networks by two mechanisms: Address space isolation among virtual networks, which ensures VMs only access VMs that belong to the same virtual network; and virtual network resources sharing isolation, which prevents virtual networks against using resources of other virtual networks [22]. The system also offers Quality of Service through mapping parameters of Service Level Agreements, defined as control plane directives, to parameters of the data plane. It controls the basic resources of virtual networks: processing, memory, and bandwidth, as those are the resources that can be locally controlled [23].

The XenFlow routing function is performed by a flow table dynamically controlled by POX, an OpenFlow network controller [24]. Migration of virtual routers, shown in Figure 1.5, consists of three steps: migration of control plane, reconstruction of data plane, and migration of virtual links. The control plane is migrated between two physical network nodes through the live-migration mechanism of conventional Xen virtual machines [15]. Then, the reconstruction of data plane is performed as follows. The virtual router sends all routes to the Domain 0. When the virtual router detects a connection disruption caused by the migration, it reconnects to the Domain 0 in new physical host and sends all information about the routing and ARP tables. Upon receiving such information, Domain 0 reconfigures the data plane according to the control plane of the migrated virtual router. After migration of the control plane and reconstruction of the data plane, links are migrated. The links migration occurs in the OpenFlow switches instantiated in Domain 0 and other OpenFlow hardware switches. Link migration creates a switched path between the neighbors of the migrated virtual router to the physical host of virtual router after migration. The migrated virtual router sends an `ARP reply` packet with a predefined destination MAC address (`AA:AA:AA:AA:AA:AA`), which the network controller captures and reconfigures the paths. This procedure updates the location of a virtual router after the migration procedure, hence, the source physical host forward packets until the migration is complete, which results in a migration primitive of virtual routers without packet loss or interruption of packet-forwarding services.

XenFlow ensures the virtual router migration without packet loss, but the new path in the underlying substrate may introduce a greater or a smaller delay when compared to the original path. XenFlow does not control delay in forwarding nodes and also the new path may comprise non-XenFlow nodes. Therefore, during virtual network migration, packets may be out of order or may be received after a bigger delay of the new path. We assume that this is not a constrain because transport protocols are resilient to delay variation, as currently occurs due to changes in routing path or network congestion.

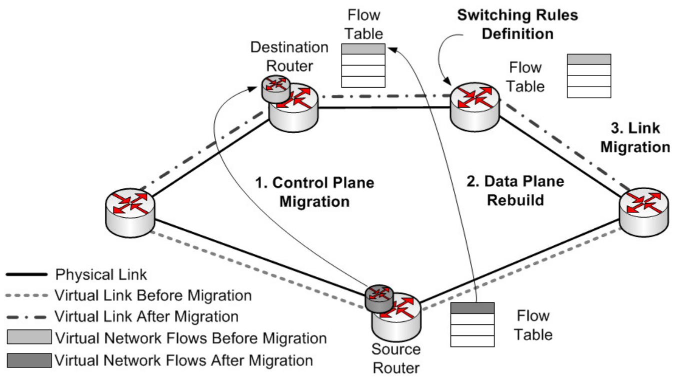

Flow Table
Switching Rules Definition
Destination Router
Flow Table
3. Link Migration
1. Control Plane Migration
2. Data Plane Rebuild
Physical Link
Virtual Link Before Migration
Virtual Link After Migration
Virtual Network Flows Before Migration
Virtual Network Flows After Migration
Source Router
Flow Table

Figure 1.5 XenFlow virtual topology migration. 1) Virtual machine and all running routing protocol migration. 2) Data plane reconstruction based on control plane information. 3) Link migration by sending a predefined `ARP Reply` message.

1.4 Security of Virtual Environments

There are several vulnerabilities that are disclosed in the current implementation of live migration of well know hypervisors, such as Xen and VMWare [25]. The biggest issue is that transferred data is not encrypted during migration procedure. Kernel memory, application state, sensitive data such as passwords and keys, and other migration data are transferred clearly, resulting in no confidentiality. Other vulnerabilities are: no guarantees that the VM is migrating to a trusted destination platform, no authentication and no authorization of operations, no integrity guarantees of VM data, and bugs in the hypervisor and migration module code that introduce security vulnerabilities. In this section, we argue about the main security issues of machine virtualization and we expose the main security requirements for a secure virtualization platform. We focus on securing virtual machine migration, but we also highlight security issues that affect Cloud Computing environment based on machine virtualization.

1.4.1 Requirements for a Secure Virtual Environment

A secure virtualization environment must ensure that processor, RAM, storage, and network, the main resources of a virtual machine, are invulnerable against other virtual machines or against infrastructure attacks. Therefore, we establish six security requirements that summarize the needs of a secure virtualization environment. We also highlight that a secure live migration should provide confidentiality, to guarantee that any VM data are not accessed by others while they are transferred from one host to another, and auditability, to secure that sensitive data have not been exposed or damaged [26]. The six secure virtualization requirements are following: Availability and isolation; Integrity; Confidentiality; Access Control, Authentication and Authorization; Non-Repudiation; and Replay Resistance.

Availability and Isolation stands for the fact that any virtual machine should be neither capable to access nor interfere other virtual machines. Even though several virtual machines share the same infrastructure, one virtual machine is not able to access other virtual machines data or change computing results [1]. Thus, a secure hypervisor ensures strong isolation between running virtual machines, running each virtual machine into a protected domain [27]. It is worth noting that isolation is achieved with confidentiality, integrity, and protection against denial of service.

Integrity aims that a virtual environment must provide the means to verify and prove the integrity and, therefore, it must be possible to identify if its processing, memory and storage were modified. Attacks against integrity intend to modify information from virtual environments or to modify running programs in a virtual environment. The migration process should also be protected against integrity violation, because it clearly exposes the virtual machine memory through the network to attacks, such as man in the middle attack [28]. In addition, a hardware module can run cryptographic functions to perform integrity verification and attestation. Attestation cryptographically ensures that a computing environment is trustworthy and the running application are not compromised [27]. Attestation may also assure that a remote environment is trustworthy because it has the same cryptographic signature of an integer environment. Attestation is also important to assure that after a virtual machine migration, the destination machine is trustworthy and the migrated virtual machine keeps its integrity as its cryptographic signature remains the same of the one before migration.

VM atomicity ensures that only one instance of the VM runs at a time [10]. Therefore, VM migration should neither add new VMs nor eliminate anyone. Thus, after successful migration, the system removes the VM instance in source host, and in case of migration failure, the system removes the VM instance in target host. The atomicity is crucial to ensure the integrity of the infrastructure for disaster recovery and to avoid generating duplicated copy of the same virtual machine.

Confidentiality ensures that an attacker should not be able to intercept, to access or to modify the content of data transfer during the migration of a virtual machine. Therefore, system may use secure communication channel to transfer data between peer hosts Moreover, the peers of the secure communication channel should be able to negotiate unique cryptographic keys and ensure that they are known only by the peers [10].

Access Control, Authentication and Authorization define that the system must ensure that a VM migration is performed between two secure authenticated platforms, which both are authorized to perform the migration, and there is no one else between them (man in the middle). Authentication ensures the true identity of an entity, hence, other security requirements depend on successful authentication. Authentication is a key feature because other security requirements depend on the authentication such as authorization, to distinguish legitimate and authorized from illegitimate participants based on authentication. Authorization ensures that only authorized entities perform operations such as VM migration. Besides, the VM should be neither migrated to unauthorized host nor from one [29].

Non-repudiation stands for the peers involved in migration cannot deny the migration participation [10]. The system must guarantee the provision of conclusive evidences of the migration event and peers participation, even when peers do not cooperate.

Replay resistance aims that an attacker cannot reproduce the migration procedure without being detected. Hence, all migration packets are unique and lose validity after migration.

1.4.2 Vulnerabilities

In a virtual environment, multiple virtual machines running on top of the same physical machine increase the efficiency of the system, but it also introduces software on sensitive areas of the system, which increases vulnerabilities. These vulnerabilities can be exploited by malicious users to obtain sensitive information, such as passwords and encryption keys, or perform other types of attacks such as denial of service.

In internal attacks, the system administrator performs attacks on the virtual machines. In this case, the system is completely vulnerable, because the administrator is authenticated and authorized to perform actions, neither cryptographic nor integrity techniques prevent the attacks. A malicious user who gains super-user privileges, via flaws in the authentication and authorization modules, performs an internal attack.

In other attacks, the attacker exploits the flaws of the virtualization system source code to inject malicious code and modify the system modules. This attack is possible due to the complexity of virtualization systems that end up having security flaws [2].

The attack can also be originated from an infected virtual machine (or a legitimate machine with a malicious user) targeting other virtual machines sharing the same system. This type of attack requires that the attacker and the target virtual machine are in the same physical machine. Due to the sharing of resources (CPU data cache, e.g.), the attacker can steal cryptographic keys using techniques such as covert channel. This attack is facilitated when the network infrastructure indirectly allows the user to map the virtual networks and verify co-residence with the target virtual machine [1]. These procedures are facilitated when static IPs are used for virtual networks, associating them with the physical IPs, but it can also be checked with IP common tools, such as `traceroute`.

The side channel attack is any attack that information, obtained to break the system, relies on information leaked by the hardware that are obtained by physical measurements as a "side" or an alternative channel [30]. The attack only concerns the implementation of a cryptosystem, rather than cryptanalysis of the math of the algorithm or brute force. Examples of physical measurements used to build a side channel can be: time took for performing different computations [31], varying power consumption [32] or leaked electromagnetic radiation provided by the hardware during computations, and even sound produced by the hardware. Therefore, assuming side attacks, the weakness of the security system is not the algorithm but its implementation. Brumley and Boneh [33] have shown that they succeeded to extract private keys from an OpenSSL-based web server running on a machine in the local network. They run a timing attack in which an attacker machine measures the decryption queries response time of an OpenSSL server, in order to extract the private key stored on the server. They successfully performed the attack between two virtual machines, then, their results invalidate the announced isolation provided by the hypervisor. As mentioned before, side channel attacks only concern the crypto algorithm implementation and, thus, a virtualized system does not interfere on the weakness or strengthen of an implementation. Otherwise, virtualization is a shared operating hardware environment and actions of one virtual machine may cause effects in another virtual machine. Therefore, a virtualized system should not facilitate the access to physical measurements and should fully isolate one virtual environment from another virtual environment to prevent side channels attacks.

Covert channel is a type of security attack that creates and conveys information through a hidden communication channel, which is able to transfer information between processes that violate the security policy. A covert channel is not a legitimate channel and, therefore, it depends upon an ingenious mechanism, which is a program scheme to hide the way used to transfer the information from the source to the destination and requires access to the file system. Hence, different from side channel attack, covert channel are illegitimate communication channel built on already compromised systems. Covert channel requires viral infection of the system or a programming effort accomplished by the administrator or other authorized user of the system. Covert channels are usually difficult to detect and low detectability, the capacity to stay hidden, is often the assumed measurement of effectiveness of a covert channel attack. The usual hardware based security mechanisms that underlie ultra-high-assurance secure operating systems cannot detect or control covert channels because they do not employ legitimate data transfer mechanisms of the computer system such as read and write. Thus, the covert channel must not interfere into legitimate operations to not be detected by security systems.

Intruders have limited options to get the data out of secured systems with Intrusion Detection Systems, Packet Anomaly Detection systems, and firewalls [34]. In this scenario, the intruder creates a covert channel. The communication media often used are ordinary actions unnoticed by administrator and legitimate users such as use of header- or payload-embedded information, altering a store location, performing operations that modify the real response time, using of packet inter-arrival times, etc. Adding data to the payload section of Ping packets or encoding data in the unused fields of packet headers. A covert channel attack, which is the most difficult to detect, is to use inter-packet delay times to encode data. This means that the intruder does not necessarily have to create new traffic because he encodes the data by modulating the time between packets of regular legitimate communication. Data exfiltration can be an indication that a computer has been compromised even when other intrusion detection schemes have failed to detect a successful attack.

During the process of live migration, vulnerabilities may be exploited by attackers. Such vulnerabilities include authorization, integrity and isolation failures.

Inappropriate access control policy: If access control policies are not defined properly or the module responsible for regulating them does not act effectively, an attacker can acquire undue control of systems to perform internal attacks. When the attacker controls the migration operation, the attacker can cause a denial of service by migrating multiple VMs to one physical machine to overload the communication link and the physical machine itself. The attacker may also migrate a malicious virtual machine to a target physical machine, or migrating a target virtual machine to a malicious physical machine. In both cases, after migration, the attacker gains full control of the target machine (physical or virtual).

Unprotected channel transmission: If the migration channel does not guarantee the confidentiality of the data, an attacker can steal or modify sensitive information, such as passwords and encryption keys. Attacks can be done passively (sniffing) or actively (man-in-the-middle) using techniques such as ARP spoofing, DNS poisoning and route hijacking. Active attacks are usually more problematic since they violate integrity, and may include modifications in the authentication services of the virtual machine (sshd/login) and manipulation of kernel memory.

Loopholes in the migration module: The contemporary virtualization software such as Xen, VMware and KVM, have an extensive and complex code base, which tend to have bugs. Perez-Botero *et al.* identified 59 vulnerabilities in Xen and 38 in KVM until July 15, 2012, according to reports of CVE security vulnerability database [35]. These results confirm the existence of vulnerabilities, which an attacker can exploit to obstruct or access virtual machines.

1.4.3 Isolation, Access Control, and Availability

Several proposals aim to improve virtualization isolation, QoS provisioning, and virtual topologies migration. Besides, some proposals use Software Defined Networking (SDN) to manage network migrations. There are proposals for developing security applications on OpenFlow network infrastructures, as there are others that seek to ensure the security of the infrastructure itself [36].

NetLord [37] introduces a software agent on each physical server, which encapsulates packets of virtual machines with a new IP header. The new IP header whose semantics of addresses of Layers 2 and 3 are overloaded to indicate to which virtual network the frames belong to. Similarly, VL2 [38] encapsulates IP packets of a virtual network with another IP header. In this case, the semantics of the IP addresses indicate both the virtual network and the localization of the physical host.

Distributed Overlay Virtual Ethernet (DOVE) [39] is a proposal of network virtualization that provides address space isolation by using a network identifier field of the envelop DOVE header, creating an overlay network. Address space isolation is also achieved using VXLAN encapsulation [40]. VXLAN also adds to each Ethernet frame an outer Ethernet header, followed by an external IP, UDP and VXLAN headers. *Network Virtualization Generic Routing Encapsulation* (NVGRE) [41] also encapsulates to allow multi-tenancy in public or private clouds. Both VXLAN and NVGRE use 24 bits to identify the virtual network that a frame belongs to. Nevertheless, these proposals create an overlay network that interconnects the nodes of the virtual network.

Houidi *et al.* propose an adaptive system that provides resources on demand for virtual networks [42]. It provides more resources for virtual networks as soon it detects service degradation or after a resource failure. The system uses a distributed multi-agent mechanism in physical infrastructure to negotiate requests, to fit the resources to the network needs, and to synchronize supplier nodes and virtual networks. Another proposal, OMNI (OpenFlow Management Infrastructure) [43] provides Quality of Service (QoS) to OpenFlow networks [21]. OMNI manages all flows of the network and define QoS parameters to each one. Besides, OMNI migrates flows to different paths without any packet losses. Kim *et al.* map QoS parameters of the virtual networks with different workloads on resources available on OpenFlow switches, such as queues and rate limiters [44]. The proposal main goal is to provide QoS to scenarios in which the physical infrastructure belongs to a cloud multi-tenant provider. Nevertheless, the control of QoS parameters and QoS mapping are centralized on the OpenFlow controller node. McIlroy and Sventek provide QoS to virtual networks with a new router architecture [45]. The router is composed of multiple virtual machines, called *Routelets*. Each it Routelet is isolated from others and their resources are limited and guaranteed. it Routelets who route QoS sensitive flows have access priority to substrate resources. Nevertheless, packet forwarding is performed by virtual machines, which limits the forwarding performance of it Routelets.

Wang *et al.* propose a load balancer based on programming low cost OpenFlow switches to multiplex requests among different server replicas [46]. The proposed solution weightily fragments the IP address space of clients between server replicas. Thus, according to the client IP, it identifies the replica that serves a client. The proposal, however, does not guarantee the reservation of resources, nor QoS of flows. Hao *et al.* present the infrastructure VICTOR (*Virtually Clustered Open Router*) which is based on creating a cluster of datacenters via a virtualized network infrastructure [47]. The central idea of this approach is to use the OpenFlow as the basic network infrastructure of datacenters to allow moving a virtual machine from one physical location to another, as it is possible to reconfigure network paths. This proposal optimizes the datacenter network usage performing server migrations, but it does not guarantee Quality of Service of each flow, and also does not isolate the use of resources from different virtualized servers.

1.5 Future Directions

The most important performance goal in Virtual Machine live migration is a short VM downtime. Current migration approaches apply a combination of push and stop-and-copy strategies for VM live migration. The combined push and stop-and-copy strategy reduces the VM downtime at the cost of increasing the total migration time and network traffic due to migration. When transferring the VM storage during migration, total migration time is also affected. Therefore, a main research topic is to decrease the total downtime, keeping memory and storage consistence and reducing network bandwidth. Downtime directly impacts on the virtualization performance and compromises the deployment of virtual machine migration on different scenarios.

Virtual Network Migration is another research topic. When moving a virtual machine, its network connections should follow accordingly. VM migration between different Local Area Networks demands mechanisms for IP address migration or for networks traffic redirection. Migration within the same datacenter can also present performance problems when datacenters are globally distributed in a wide geographical area. Current research efforts focus on tunneling network traffic between source and destination host [22]. In this direction, there are proposal, such as NVGRE (Network Virtualtization Generic Routing Encapsulation) [41], VXLAN (Virtual eXtended Local Area Network) [40], and DOVE (Distributed Overlay Virtual Ethernet) [39], that creates tunnels to maintain virtual network connectivity even in scenarios that sites are separated by a Wide Area Network. Moreover, NetLord [37] and VL2 [38] change IP semantics for isolating and creating virtual networks within a datacenter. Other proposals for handling virtual machine mobility across the Internet is to use Locator/Identifier Separation Protocol (LISP) [48, 49]. LISP uses two IP headers, one for the locator and other for identifier of the host. LISP maintains a globally reachable service that maps locator into identifier, and vice-versa, in order to ensure the correct location of virtual machine no matter where it is hosted. After the virtual machine migration, only the locator is changed, and all services remain online and reachable. Future trends also point to OpenFlow [21] as a possible approach for managing virtual network. Nevertheless, all above mentioned approaches require adaptations or more sophisticated deployments to be fully functional. To achieve a seamless network migration, we believe that new standards should take place to define a common way to migrate virtual network.

Storage is an important resource to virtualized servers, because it must be always available and present high performance. When migrating a VM, its storage should be also available at migration destination. Therefore, both source and destination sites share the storage service, or all VM storage must be sent over the network to destination host. EMC^2, one of the world's leader storage provider enterprise, provides a storage facility focused on a distributed federation of data, which allows data to be accessed among locations over synchronous distances. The EMC^2 distributed storage service is called VPLEX [2]. Moreover, Ceph is an open source project which aims to provide a distributed and redundant file system [50]. We agree that there are several initiatives for providing distributed storage service that are a step ahead for an available file system for virtual machine migration. Nevertheless, these initiatives are new and immature. The proprietary ones have a higher maturity grade, but still are expensive and demand large infrastructure. Providing a distributed and available storage service, that requires low investment into infrastructure and is backward compatible, is a key research area.

[2]http://www.emc.com/campaign/global/vplex/index.htm.

Automated migration is also a key research topic, because the virtual machines allocation into physical servers is a np-hard problem. This scenario is aggravated considering big datacenters and multiple datacenters in a Cloud Provider environment, due to the size and unmanageability of the scenario. There are proposed heuristics [51] based optimization and others based on system modeling [52, 53], aiming to better use physical resources. An important factor to be considered in the use of optimization algorithms is the convergence time of the algorithm, which will directly interfere into the dynamics of the system. Proposals for optimization of use of physical resources are complementary to automatic migration systems and can be used to manage migrations. Trends show that a key research theme is matching the tradeoff between optimizing physical resource usage and limiting the number of migrations into the network.

A major research topic that arises is securing virtual machine migration. Our studies show that there is no proposal that achieves a complete secure live migration primitive. Security must be deployed all long the development of a virtualization system. It must be present since the hypervisor, which should be reliable, trustworthy, and should provide secure virtualized environment, till the migration procedure, which should authenticate peers, check the trust of the foreign peer, and ensure a confidential channel between peers for transferring the virtual machine. Security must also be ensured for all resources used by a virtual machine. Isolation is a key challenge for network virtualization, as availability is another key challenge for storage virtualization. Confidentiality is an open topic while virtualizing memory. Trust warranting is a trend of research, in which we identified some works proposing protocols and new approaches [54]. We believe that providing security for virtualized environments is a hot research topic, in which the proposals still are initial and immature. Therefore, trends show that new security mechanisms should be proposed for guaranteeing a securer virtualizing system.

1.6 Conclusion

Virtual Machine Migration is one of the most useful primitive introduced by virtualization technique. Virtual Machine Migration stands for the relocation of virtual computing environments over the physical infrastructure. The main idea of the migration primitive is to remap virtual resources into physical resources without disrupting the function of the virtual resources. We consider Virtual Machine Migration of particular interest for cloud computing environments and for network virtualization approaches. We claim that migration is a powerful tool for fitting computer capacity to dynamic workloads, facilitating user mobility, improving energy savings, and managing failures. In a network virtualization scenario, virtual machine migration plays the hole of flexibly changing network topologies without constraining the physical realization of the virtual topology. Nevertheless, virtual machine migration is both challenging in its realization and in its security guarantees.

In this chapter, we explained that live migration is the key migration mechanism of most of current hypervisor. We identified that the key resource to migrate is the virtual machine memory, as it is constantly updated during the migration process. We also discussed how to migrate storage service of virtual machines through wide area networks. Moreover, we present a network virtualization approach, called XenFlow, which focuses on migrating virtual networks, without losing packets or disrupting network services. Besides the technical difficulties of migrating a virtual machine, while it is running, we also highlighted how to assure that a virtual machine migration occurs in a secure environment.

REFERENCES

1. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, “Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds,” in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS ’09. New York, NY, USA: ACM, 2009, pp. 199–212. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653687

2. Z. Wang and X. Jiang, “Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity,” in *Security and Privacy (SP), 2010 IEEE Symposium on*, May 2010, pp. 380–395.

3. M. Pearce, S. Zeadally, and R. Hunt, “Virtualization: Issues, security threats, and solutions,” *ACM Comput. Surv.*, vol. 45, no. 2, pp. 17:1–17:39, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2431211.2431216

4. Z. Pan, Y. Dong, Y. Chen, L. Zhang, and Z. Zhang, “Compsc: Live migration with pass-through devices,” *SIGPLAN Not.*, vol. 47, no. 7, pp. 109–120, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2365864.2151040

5. L. H. G. Ferraz, D. M. F. Mattos, and O. C. M. B. Duarte, “A two-phase multipathing scheme with genetic algorithm for data center network,” IEEE Global Communications Conference - GLOBECOM (to appear), December 2014.

6. O. C. M. B. Duarte and G. Pujolle, *Virtual Networks: Pluralistic Approach for the Next Generation of Internet*. John Wiley & Sons, 2013.

7. I. M. Moraes, D. M. Mattos, L. H. G. Ferraz, M. E. M. Campista, M. G. Rubinstein, L. H. M. Costa, M. D. de Amorim, P. B. Velloso, O. C. M. Duarte, and G. Pujolle, “FITS: A flexible virtual network testbed architecture,” *Computer Networks*, no. 0, pp. –, 2014.

8. H. T. Mouftah, H. T. Mouftah, and B. Kantarci, *Communication Infrastructures for Cloud Computing*, 1st ed. Hershey, PA, USA: IGI Global, 2013.

9. S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vtpm: Virtualizing the trusted platform module," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. USENIX Association, 2006.

10. X. Wan, X. Zhang, L. Chen, and J. Zhu, "An improved vtpm migration protocol based trusted channel," in *Systems and Informatics (ICSAI), 2012 International Conference on*, May 2012, pp. 870–875.

11. M. Aslam, C. Gehrmann, and M. Bjorkman, "Security and trust preserving VM migrations in public clouds," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, June 2012, pp. 869–876.

12. D. M. F. Mattos and O. C. M. B. Duarte, "XenFlow: Seamless migration primitive and quality of service for virtual networks," IEEE Global Communications Conference - GLOBECOM (to appear), December 2014.

13. Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 231–242. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402985

14. P. Pisa, N. Fernandes, H. Carvalho, M. Moreira, M. Campista, L. Costa, and O. Duarte, "Openflow and xen-based virtual network migration," in *Communications: Wireless in Developing Countries and Networks of the Future*, ser. IFIP Advances in Information and Communication Technology, A. Pont, G. Pujolle, and S. Raghavan, Eds. Springer Boston, 2010, vol. 327, pp. 170–181.

15. C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

16. S. Venkatesha, S. Sadhu, and S. Kintali, "Survey of virtual machine migration techniques," Department of Computer Science - University of California, Santa Barbara, CA, U.S.A, Tech. Rep., Mar. 2009.

17. J. Suzuki, Y. Hidaka, J. Higuchi, T. Baba, N. Kami, and T. Yoshikawa, "Multi-root share of single-root i/o virtualization (sr-iov) compliant pci express device," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, Aug 2010, pp. 25–31.

18. E. Zhai, G. D. Cummings, and Y. Dong, "Live migration with pass-through device for linux vm," in *OLS'08: The 2008 Ottawa Linux Symposium*, 2008, pp. 261–268.

19. N. Fernandes, M. Moreira, I. Moraes, L. Ferraz, R. Couto, H. Carvalho, M. Campista, L. Costa, and O. Duarte, "Virtual networks: Isolation, performance, and trends," *Annals of Telecommunications*, pp. 1–17, 2010.

20. N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Towards high performance virtual routers on commodity hardware," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, pp. 1–12.

21. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev., 2008*, Mar. 2008.

22. M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, 2013.

23. R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," Tech. Rep. OPENFLOW-TR-2009-01, OpenFlow Consortium, Tech. Rep., 2009.

24. M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, “Virtualizing the network forwarding plane,” in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*. ACM, 2010, p. 8.

25. V. Melvin, “Dynamic load balancing based on live migration of virtual machines: Security threats and effects,” Master’s thesis, B. Thomas Golisano College of Computing and Information Sciences (GCCIS) - Rochester Institute of Technology, Rochester, NY, U.S.A, 2011.

26. Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010. [Online]. Available: http://dx.doi.org/10.1007/s13174-010-0007-6

27. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, “Terra: A virtual machine-based platform for trusted computing,” in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ser. SOSP ’03. New York, NY, USA: ACM, 2003, pp. 193–206. [Online]. Available: http://doi.acm.org/10.1145/945445.945464

28. J. Oberheide, E. Cooke, and F. Jahanian, “Empirical exploitation of live virtual machine migration,” in *Proc. of BlackHat DC convention*, 2008.

29. B. Danev, R. J. Masti, G. O. Karame, and S. Capkun, “Enabling secure VM-vTPM migration in private clouds,” in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC ’11. New York, NY, USA: ACM, 2011, pp. 187–196. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076759

30. D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, “The EM side—channel(s),” in *Cryptographic Hardware and Embedded Systems - CHES 2002*, ser. Lecture Notes in Computer Science, B. S. Kaliski, c. K. Koç, and C. Paar, Eds. Springer Berlin Heidelberg, 2003, vol. 2523, pp. 29–45. [Online]. Available: http://dx.doi.org/10.1007/3-540-36400-5_4

31. P. C. Kocher, “Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems,” in *Advances in Cryptology — CRYPTO ’96*, ser. Lecture Notes in Computer Science, N. Koblitz, Ed. Springer Berlin Heidelberg, 1996, vol. 1109, pp. 104–113. [Online]. Available: http://dx.doi.org/10.1007/3-540-68697-5_9

32. P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Advances in Cryptology — CRYPTO’ 99*, ser. Lecture Notes in Computer Science, M. Wiener, Ed. Springer Berlin Heidelberg, 1999, vol. 1666, pp. 388–397. [Online]. Available: http://dx.doi.org/10.1007/3-540-48405-1_25

33. D. Brumley and D. Boneh, “Remote timing attacks are practical,” *Computer Networks*, vol. 48, no. 5, pp. 701 – 716, 2005, web Security. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128605000125

34. C. Fung, D. Lam, and R. Boutaba, “RevMatch: An Efficient and Robust Decision Model for Collaborative Malware Detection,” in *IEEE/IFIP Network Operation and Management Symposium (NOMS14)*, 2014.

35. D. Perez-Botero, J. Szefer, and R. B. Lee, “Characterizing hypervisor vulnerabilities in cloud computing servers,” in *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, ser. Cloud Computing ’13. New York, NY, USA: ACM, 2013, pp. 3–10. [Online]. Available: http://doi.acm.org/10.1145/2484402.2484406

36. D. Kreutz, F. M. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN ’13. New York, NY, USA: ACM, 2013, pp. 55–60.

37. J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, “Netlord: a scalable multi-tenant network architecture for virtualized datacenters,” in *Proceedings of the ACM SIGCOMM 2011*, ser. SIGCOMM ’11. Toronto, Ontario, Canada: ACM, 2011, pp. 62–73.

38. A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “Vl2: a scalable and flexible data center network,” in *Proceedings of the ACM SIGCOMM 2009*, ser. SIGCOMM ’09. New York, NY, USA: ACM, 2009, pp. 51–62.

39. K. Barabash, R. Cohen, D. Hadas, V. Jain, R. Recio, and B. Rochwerger, “A case for overlays in DCN virtualization,” in *Proceedings of the 3rd Workshop on Data Center - Converged and Virtual Ethernet Switching*, ser. DC-CaVES ’11. ITCP, 2011, pp. 30–37.

40. Y. Nakagawa, K. Hyoudou, and T. Shimizu, “A management method of IP multicast in overlay networks using openflow,” in *Proceedings of the first workshop on Hot topics in software defined networks*, ser. HotSDN ’12. Helsinki, Finland: ACM, 2012, pp. 91–96. [Online]. Available: http://doi.acm.org/10.1145/2342441.2342460

41. M. Sridharan, K. Duda, I. Ganga, A. Greenberg, G. Lin, M. Pearson, and P. Thaler, “NVGRE: Network Virtualization using Generic Routing Encapsulation,” NVGRE, Internet Engineering Task Force, Feb. 2013. [Online]. Available: http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-02

42. I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, “Adaptive virtual network provisioning,” in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*. ACM, 2010, pp. 41–48.

43. D. M. F. Mattos, N. C. Fernandes, V. T. da Costa, L. P. Cardoso, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, “OMNI: Openflow management infrastructure,” in *Network of the Future (NOF), 2011 International Conference on the*. IEEE, 2011, pp. 52–56.

44. W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S. Lee, and P. Yalagandula, “Automated and scalable QoS control for network convergence,” *Proc. INM/WREN*, Apr. 2010.

45. R. McIlory and J. Sventek, “Resource virtualisation of network routers,” in *High Performance Switching and Routing, 2006 Workshop on*. IEEE, 2006, pp. 6–pp.

46. R. Wang, D. Butnariu, and J. Rexford, “Openflow-based server load balancing gone wild,” in *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*. USENIX Association, 2011, pp. 12–12.

47. F. Hao, T. Lakshman, S. Mukherjee, and H. Song, “Enhancing dynamic cloud-based services using network virtualization,” in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. ACM, 2009, pp. 37–44.

48. D. Phung, S. Secci, D. Saucez, and L. Iannone, “The openlisp control plane architecture,” *Network, IEEE*, vol. 28, no. 2, pp. 34–40, March 2014.

49. P. Raad, S. Secci, D. C. Phung, A. Cianfrani, P. Gallard, and G. Pujolle, “Achieving subsecond downtimes in large-scale virtual machine migrations with lisp,” *Network and Service Management, IEEE Transactions on*, vol. 11, no. 2, pp. 133–143, Jun. 2014.

50. S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, “Ceph: A scalable, high-performance distributed file system,” in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, ser. OSDI ’06. Berkeley, CA, USA: USENIX Association, 2006, pp. 307–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298455.1298485

51. I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, “Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic,” in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1–6.

52. E. Rodriguez, G. Alkmim, D. Batista, and N. da Fonseca, “Live migration in green virtualized networks,” in *Communications (ICC), 2013 IEEE International Conference on*, 2013, pp. 2262–2266.

53. G. P. Alkmim, D. M. Batista, and N. L. S. da Fonseca, “Mapping virtual networks onto substrate networks,” *Journal of Internet Services and Applications*, vol. 4, no. 1, 2013. [Online]. Available: http://dx.doi.org/10.1186/1869-0238-4-3

54. L. H. G. Ferraz, P. B. Velloso, and O. C. M. Duarte, “An accurate and precise malicious node exclusion mechanism for ad hoc networks,” *Ad Hoc Networks*, vol. 19, no. 0, pp. 142 – 155, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870514000468