Network Function Virtualization: Challenges and Directions for Reliability Assurance

D. Cotroneo, L. De Simone, A.K. Iannillo, A. Lanzaro, R. Natella
Critiware s.r.l. / Federico II University of Naples, Italy

Jiang Fan, Wang Ping
Huawei Technologies Co., China

***Abstract*—Network Function Virtualization (NFV) is an emerging solution that aims at improving the flexibility, the efficiency and the manageability of networks, by leveraging virtualization and cloud computing technologies to run network appliances in software. Nevertheless, the "softwarization" of network functions imposes software reliability concerns on future networks, which will be exposed to software issues arising from virtualization technologies. In this paper, we discuss the challenges for reliability in NFVIs, and present an industrial research project on their reliability assurance, which aims at developing novel fault injection technologies and systematic guidelines for this purpose.**

***Keywords*—NFV; Fault Injection Testing; Cloud Computing; Virtualization**

I. Introduction

The landscape of communication networks presents many different actors such as cloud service providers, enterprise networks, content delivery networks, mobile users, which are demanding more and more performance, reliability and security. To meet these needs, modern networks deploy a wide range of network appliances (*middleboxes*) to provide advanced services such as intrusion detection and prevention systems, application-level firewalls and gateways, traffic shapers, and several others [1]. From one side, middleboxes introduce valuable benefits in term of provided functionalities, but from the other side they constitute an important fraction of the OPerational EXpenditures (OPEX) and CAPital EXpenditures (CAPEX) for telecom operators. In fact, middleboxes are usually based on proprietary hardware and software, and are tailored only for some specific function. Thus, middleboxes are costly, have limited flexibility, are energy-inefficient, are difficult to manage and to troubleshoot [2], and their failures have a strong impact on network performance and availability [3].

Network Function Virtualization (NFV) [4], [5] is an emerging solution to overcome these problems. According to the ETSI Industry Specification Group for NFV, established by leading telecoms network operators, NFV exploits IT virtualization technologies to turn network equipment (i.e., middleboxes) into virtual entities. *Virtualized Network Functions* (VNFs)[1] will be implemented in software and will run on commodity hardware located in already-existing data centers, network nodes and even in end-user premises. By doing that, network operators can reduce costs, improve efficiency, reduce time-to-market, and provide more advanced services [6].

[1]NFV indicates the technology for virtualizing network functions, VNF indicates the virtual entity that performs a network function.

Beyond others, cloud computing technologies are the most important enablers for NFV, and represent a critical block of the *NFV infrastructure* (NFVI) on which VNFs are deployed. Virtualization technologies, such as hypervisors and containers, allow to abstract physical computing resources (e.g., CPUs, network and storage devices) in order to achieve efficiency and elasticity (e.g., by dynamically allocating resources to VNFs), and to easily manage and orchestrate VNFs throughout their lifecycle (creation, deletion, migration, etc.).

It can be easily seen that the "softwarization" of network functions imposes software reliability concerns on future networks. While off-the-shelf hardware components are expected to fail and to be easily replaced, with very low configuration or management efforts, software (and, in particular, virtualization technologies) will represent the weak point for NFV, raising new questions like:

- *What are the risks of leveraging on virtualization technologies in NFV infrastructures?*
- *How can we predict and mitigate the impact of faults arising from virtualization technologies?*

The European Union, to meet the needs of telecom operators and service providers, added the certification of security and reliability of cloud systems among the high-priority topics of the Horizon 2020 research program [7]. The aim is to encourage the development of proof-of-concepts, best practices, test suites and benchmarks to assure cloud resiliency.

In this paper, we discuss the challenges for reliability in NFVIs, and present an industrial research project on their reliability assurance, which aims at developing novel fault injection technologies and systematic guidelines for this purpose. The paper is organized as follows: Section II provides background on NFV; Section III details the context, objectives and challenges of our reliability evaluation project; Section IV discusses existing fault injection tools and techniques for cloud systems; Section V closes the paper with future directions of the project.

II. NFV Background

A. Use Case Scenarios

In principle, all network functions and nodes may be considered for virtualization but, in order to span the scope of technical challenges, NFV ISG selected a set of relevant use case scenarios [8], such as:

- Network Functions Virtualization as a service: NFV infrastructure, platform and even a single VNF instance can be provided as a service by a Service

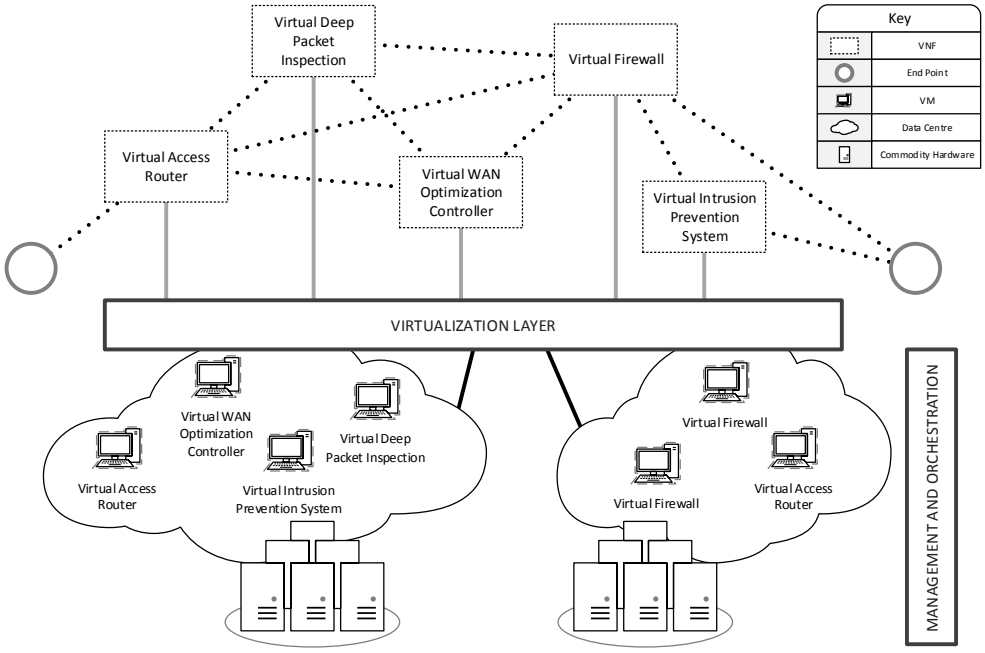

Fig. 1. VNF-FG scenario

Provider, based on models similar to the cloud computing service models [9];

- Virtualization of Mobile Core Network and IMS: the mobile networks and the IP Multimedia Subsystems are populated with a large variety of proprietary hardware appliances, which costs and complexity can be reduced introducing NFV;
- Virtualization of Mobile base station: mobile operators can apply NFV in order to reduce costs as well as continuously develop and provide better service to their customer;
- Virtualization of the Home Environment: Installation of new equipment can be avoided in the home environment with the introduction of VNFs, reducing maintenance and improving service provision;
- Virtualization of CDNs: Content Delivery Networks use cache node to improve the quality of multimedia services, but it comes with lots of disadvantages (e.g., waste of dedicated resources) that could be mitigated by NFV;
- Fixed Access Network Functions Virtualization: virtualization supports multiple tenancy in access network equipment, whereby more than one organizational entity can either be allocated, or given direct control of, a dedicated partition of a virtual access node.

In all the scenarios, service providers run VNF instances inside an *NFV Infrastructure* (NFVI)[2]. It provides the capability or functionality of an environment in which both virtualized and non-virtualized network functions can be connected into a service chain, i.e. VNF Forwarding Graph (VNF-FG)[8]. The NFVI includes common elements of cloud computing such as physical computing, network and storage resources and resource pooling mechanisms. Fig. 1 shows an example of a VNF-FG commonly encountered where packet traverse a VNF implementation of a router, a Deep Packet Inspection and a Firewall, with the possibility of adding to the service an Instruction Prevention System or a WAN Optimization Controller. Every VNF is mapped through the virtualization layer to a pool of VMs. Each VNF could be replicated on several VMs and placed on different hardware or even sites, while the **Management and Orchestration** (M&O) component decides which VNF instance a specific request should be forwarded to.

B. The NFVI Architecture

The NFV ISG is defining a potential architecture of the NFVI, to support the deployment and execution of VNFs [10].

The NFVI is part of a framework whose architecture is depicted in Fig. 2. The NFVI Domain includes the three primary domains of the NFVI, *i.e.*:

- **Compute (and Storage) Domain**: it provides the COTS computational and storage resources;
- **Hypervisor Domain**: it mediates the resources of the compute domain to the VMs of the software appliances providing an abstraction of the hardware;
- **Infrastructure Network Domain**: it provides several communication channels between entities of all the domains and it is the mean of remote deployment of VNFs.

[2]As mentioned before, a NFV is the technology for making a network function virtual, namely a VNF. The NFVI, instead, represents the environment in which more than one VNF may execute.

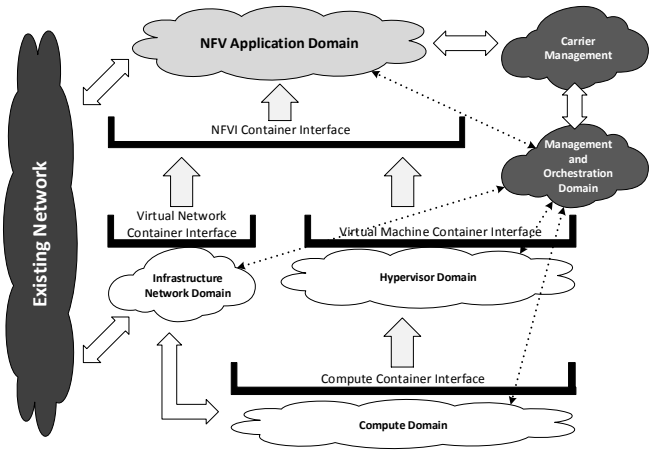

Fig. 2. NFV framework and associated interfaces

Moreover, the framework comprises the NVF Application Domain that hosts VNFs, and the M&O Domain that controls and manages software appliances running on the infrastructure.

The inter-domain communication requires a *container interface* which is the environment within a host function is configured and/or programmed in order to provide a virtualized function. The identified inter-domain interfaces are:

- **NFVI Container Interfaces**: it is provided by the infrastructure to host VNFs. The applications may be distributed and the infrastructure provides virtual connectivity which interconnects the distributed components of an application.
- **Virtual Network Container Interface**: the interface to the connectivity services, provided by the infrastructure. This container interface makes the infrastructure appear to the NFV applications as instances of these connectivity services.
- **Virtual Machine Container Interface**: it is the primary hosting interface on which the VMs run.
- **Compute Container Interface**: it is the primary compute hosting interface on which the hypervisor runs.

C. Reliability Requirements

NFV ISG has identified manifold requirements, including requirements on resiliency of VNFs [11], [12]. Telecom Operators are concerned by the availability of their products and the user-perceived dependability because (1) unreliable services are likely to be discarded by users and (2) the total costs of system failures can be tremendous. Potential causes of VNF failures are:

- **Hardware Faults**: the use of COTS servers is a source of faults in the NFVI, that may negatively affect the VNFs running on them;
- **Software Faults**: at various levels, such as host OS, hypervisor, VM, or the VNF instance itself;
- **Operator Faults**: mistaken operations and configuration, e.g. capacity planning, VM deployment and migration.

The NFV framework should be able to achieve resiliency in spite of these faults. NFV requirements will focus primarily on specific aspects introduced by NFV, and not on aspects of the network functions interfaces, protocols and management that are identical whether the implementation is physical or virtual.

The NFV framework needs to provide the necessary mechanisms to allow VNFs to be recovered after a failure. Fault-tolerance mechanisms, such as VNF redundancy, fault isolation, recovery after a failure and transparency, should guarantee the required service continuity. The NFV M&O component should be responsible for controlling these and other mechanisms, thus it is vital that it does not become a single point of failure. Above all, the NFV framework shall provide the necessary mechanisms to achieve the same level of service availability for fully and partially virtualized scenarios as for existing non-virtualized networks.

III. Reliability Evaluation Project

Our project aims to give Telecom Operators the opportunity to evaluate the reliability of an NFVI. In order to achieve this, we focus on two complementary tasks. On one hand, we will produce techniques and tools for reliability evaluation of NFVI. On the other, we will formulate methodologies and guide lines for practitioners to use these techniques and tools properly and with valuable results.

NFVI implementations will exploit the off-the-shelf products already present in the cloud-computing market, so NFVI reliability depends strictly on the reliability these products can provide. The virtualization technologies we will consider as actual enablers for NFVI are:

- **VMware vSphere** [13], a family of mature commercial products for the whole life-cycle management of hypervisor-based VMs, including high-availability mechanisms;
- **Linux Containers** [14], an emerging open-source project that enhances the Linux kernel, which runs guest applications in "containers", by abstracting hardware and OS resources (e.g., CPU time, filesystems, network interfaces, etc.) and isolating them.

In the context of business-critical scenarios, intense testing activities are of paramount importance to guarantee that new systems and their built-in fault-tolerance mechanisms are behaving as expected, and thus to assure a high level of reliability. Ensuring that the system behaves properly in the presence of a fault is a problem that requires something more than traditional testing. *Fault Injection* is the process of introducing deliberately faults in a system, with the goal of assessing the impact of faults on performance and on continuity of service, and the efficiency (*i.e.*, coverage and latency) of its fault-tolerant mechanisms.

According to resiliency and service continuity requirements of the NFV framework, as discussed in the previous sections, NFV should be able to provide mechanisms to allow network

functions to be recreated, and to assure a desired level of performance and of service continuity as mandated by SLAs. *Fault Injection Testing* is a systematic approach to assure, with quantitative evidence, that such requirements are satisfied.

Considering the architecture of NFVIs, the application of reliability evaluation approaches is not straightforward, since there are several challenges that make the NFVI reliability evaluation tool harder to design, and that will be taken into account. Within these challenges, the most influential are listed below:

- **Black-box and complex nature of virtualization technologies**: the testing and the reliability evaluation of NFVIs should be performed considering the lack of information about internal structure, design and implementation of virtualization technologies, that are often provided by third parties (*e.g.*, VMware);
- **Lack of well-established reliability evaluation criteria for NFVIs**: since NFV is a technology still under active development, it is not clear how the reliability of NFVIs should be evaluated and, thus, proper measures and metrics should be identified.
- **Integration and interoperability**: the integration and the interoperability among COTS hardware and software components increase the complexity of the infrastructure, and thus the set of faults we have to face with;

IV. Fault Injection in Cloud Computing

As mentioned in the introduction, the NFV world is strictly related to cloud computing technologies. Thus, based on *everything-as-a-service* (XaaS) delivery model, a cloud provider can develop Internet services, from security and databases to storage and integration, no longer require leverage on expensive specific-purpose hardware and on big initial capital costs.

Virtualization is an enabling technology to set up a cloud computing infrastructure. Virtualization allows to abstract physical resources (e.g., CPUs, network devices, storage devices, etc.) in order to share and to provide resources, making a physical machine as a soft component to use and manage very easily. Virtualization software is used to run one or more so-called *Virtual Machines* (VMs) (a software abstraction of a physical machine) on a single physical machine, providing the same functionalities as if they were more physical machines. This virtualization software is named *Hypervisor*, which is responsible of executing and managing multiple VMs in order to synchronize the access to the CPU, memory and other I/O resources of the physical machine.

Therefore, to assure the reliability of cloud systems, it is necessary to assess the reliability of the virtualization environment as a whole, focusing both on VMs and on the Hypervisor, as well as on the Cloud Management Stack software that orchestrates them (such as the well-known OpenStack framework [15]) to efficiently manage cloud infrastructures. In this section, we present an overview of related studies that adopt fault injection to assure a high-level of reliability of cloud systems, focusing on Virtual Machines (subsection IV-A), Cloud Management Stack (subsection IV-B), and Hypervisors (subsection IV-C).

A. Fault Injection Testing of Virtual Machine

D-Cloud [16] is a dedicated simulated test environment, based on QEMU for virtualizing physical machines, and on Eucalyptus cloud computing system for managing these VMs. D-Cloud adopts QEMU to emulate hardware faults, by injecting various typical faults into the guest OS.

DS-Bench Toolset [17] is a framework (it includes D-Cloud) that computes dependability metrics of the overall system under test (SUT), using various benchmark programs, by injecting anomaly loads; furthermore, it provides the evidence for the assurance case based on the benchmark results.

D-Cloud considers hardware faults in memory, hard-disk and network devices. It performs fault injection by simulating data corruptions in the emulated devices. The fault types include the corruption of individual sectors of the disk (e.g., the sector was damaged by head crash), of packets sent through the network (e.g., loss or bit corruption of a packet), and of memory cells. Moreover, it can simulate an unresponsive or slow hard disk and network devices.

B. Fault Injection Testing of Cloud Management Stack

A systematic study on fault resilience of **OpenStack** [15] is reported in [18]. Openstack is one of the most important open cloud computing software, that controls compute, storage and networking resources in a whole data center, managed and provisioned through a web-based dashboard, command-line tools or a RESTful API. The proposed framework injects network faults targeting communications among OpenStack's services like compute, image and identity services, but also database, hypervisor and messaging services. The authors evaluate the framework on two OpenStack versions, identifying bugs, such as timeout between services communication, or lack in period checking of service liveness (VM creation API has completed its job?) and so on, more described in the next sections.

PreFail [19] allows to deal with a very high number of injection experiments, that arises from the "combinatorial explosion" of multiple injections. The PreFail tool allows the tester to control fault injection using *pruning policies*, which select the combinations of faults to be injected during experiments. The policies offered by PreFail are oriented towards selecting a small set of faults, and to maximize the *efficiency* of fault injection tests. This goal is reached by letting the user to specify a pruning policy.

Netflix [20] is developing the **The Simian Army** [21], a set of tools (named *monkeys*) that allows to inject faults into a cloud computing platform, specifically built within AWS [22]. *Simian Army*'s "monkeys" for assessing resiliency are manifold and they allow to:

- randomly terminate virtual instances (i.e., virtual machines) in the production environment (*Chaos Monkey*);
- cause an entire data center (e.g., an Amazon availability zone (AZ)) to go down (*Chaos Gorilla*);

- bring down an entire region, made up of multiple data centers (*Chaos Kong*);
- inject faults that simulate partially healthy instances (*Latency Monkey*).

Currently, Netflix developed only the **Chaos Monkey** [23] and their related fault types. Fault injection is performed by executing a script that simulates a specific type of fault.

C. Fault Injection Testing of Hypervisor

CloudVal [24] is a framework to test the reliability of hypervisor within a cloud infrastructure. The framework provides an injector (implemented using debugger-based techniques) that allows to inject different type of faults like transient (soft) faults, guest misbehavior, performance faults and maintenance faults. This work is a starting point to develop a benchmark for validating cloud virtualization infrastructures.

The CloudVal framework supports fault injection in the KVM and Xen, both on the guest and host domains, and on the core modules of the hypervisors (i.e., *qemu-kvm* and the KVM kernel module for KVM [25]; *qemu-dm* and *xenstored* for Xen [26]). The tests are performed to evaluate VMs guest/host isolation and correlated hypervisor behavior, and the level of maintainability. Finally, *Virt-manager* [27] (a libvirt-based management system) is used by CloudVal for monitoring and managing a system during fault injection experiments.

Table I shows a comparison between fault injection approaches and the related tools mentioned above.

D. Discussion

Concerning the reliability of cloud telecom networks, the tools overviewed in this paper can be applied only to a limited extent of NFVIs since they are not designed with NFV scenarios in mind. We have identified the following limitations of existing tools, and that will need to be tackled in the development of a new Fault Injection Tool for NFVIs as mentioned in the Section III:

- **Injection of Software and Configuration faults**. So far, fault injection testing tool in cloud computing systems has mostly been focused on the injection of hardware faults (e.g. affecting CPU, memory, network and disk) to assess the tolerance and robustness of cloud systems to these faults. However, in an NFVI, the use of third-party software components, such as COTS virtualization technologies (e.g., VMware, Xen) and cloud management software (e.g., Openstack), exacerbates the overall complexity. Therefore, software and configuration faults must be included in fault injection testing, in order to predict and mitigate the impact of such faults on NFVI.
- **Black box virtualization technologies**. The surveyed tools focused on open-source virtualization technologies, such as the Xen and KVM hypervisors and the OpenStack cloud management platform. Unfortunately, performing the same on commercial off-the-shelf software (e.g., VMware, very popular in the virtualization panorama) is much more difficult, and requires additional effort to analyze the internals of the virtualization layer and to instrument it.
- **Testing scenarios for NFVIs**. NFV is a technology that is still under active development. Thus, a big challenge is to develop proof-of-concepts that could demonstrate how, in practice, Fault Injection can be applied to obtain useful measures for the NFVI architect, such as measures for benchmarking alternative components and designs for an NFVI under development.

V. Conclusion and future directions

In this paper, we presented an ongoing industrial research project, that aims at investigating how to assess the risks introduced by virtualization technologies for NFVI reliability. Towards this goal, we plan to conduct the following activities:

- **Failure Mode and Effects Analysis of virtualization technologies in NFVIs**: we need to analyze the architecture of NFVI and its potential threats in order to understand what can affect reliability. The FMEA should consider not only hardware failures, but also failures due to software and configuration faults that can impact on virtualized resources (*e.g.*, virtual CPU, memory, network and storage);
- **Definition of Key Performance Indicators and Methodologies for NFVI reliability**: we will define measures for fault tolerance and performance, and provide guidelines to allow reliability engineers to systematically assess reliability by means of fault injection testing;
- **Design of novel Fault Injection Techniques**: because of the challenges in NFVIs (e.g., black-box technologies), the most advantageous injection target seems to be represented by the interfaces of the Compute, Hypervisor and Network domains. The errors and corruptions to be injected should be defined on the basis of the FMEA;
- **Validation using NFV products and technologies**: we will conduct a proof-of-concept validation of the fault injection approach on commercial NFV products, based on virtualization technologies mentioned in the paper (*i.e.*, VMWare and LXC).

References

[1] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: Enabling innovation in middlebox deployment," in *Proc. Wksp. HotNets-X*, 2011, pp. 1–6.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem," in *Proc. SIGCOMM*, 2012, pp. 13–24.

[3] Chandler Harris. Data Center Outages Generate Big Losses. http://www.informationweek.com/data-center-outages-generate-big-losses/d/d-id/1097712.

[4] NFV ISG, "Network Functions Virtualisation - An Introduction, Benefits, Enablers, Challenges & Call for Action," ETSI, Tech. Rep., 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[5] ——, "Network Functions Virtualisation (NFV) - Network Operator Perspectives on Industry Progress," Tech. Rep., 2013. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

TABLE I. COMPARISON OF FAULT INJECTION APPROACHES.

Approach	Tool	Target	Faultload	Injection technique	Examples of results
Fault Injection Testing of Virtual Machines	D-Cloud [16] and DS-Bench Toolset [17]	Server software (e.g., web applications)	Network, disk, memory faults	Emulation of faulty devices; VM memory corruption	Validation of performance levels under faults
	Chaos Monkey [23]	Virtual instances during runtime	CPU, disk, network	Executing scripts that simulates a fault on target machine	Terminates over 65,000 instances running in Netflix production and testing environments, detecting many failure scenarios
Fault Injection Testing of Cloud Management Software	PreFail [19]	Distributed filesystems and algorithms (e.g., HDFS, ZooKeeper)	Network and disk faults; Process crashes	API exception injection	Robustness of recovery protocols
	Openstack resilience framework [18]	OpenStack	Service crash and Network partition	API exception injection	Improvement of robustness
Fault Injection Testing of Hypervisors	CloudVal [24]	Hypervisors (e.g., Xen, KVM)	CPU, memory, VM faults	Memory corruption	Improvement of VM isolation

[6] A. Manzalini, R. Minerva, E. Kaempfer, F. Callegari, A. Campi, W. Cerroni, N. Crespi, E. Dekel, Y. Tock, W. Tavernier *et al.*, "Manifesto of edge ICT fabric," in *Proc. ICIN*, 2013, pp. 9–15.

[7] European Union Agency for Network and Information Security, "Cloud computing certification." [Online]. Available: https://resilience.enisa.europa.eu/cloud-computing-certification

[8] NFV ISG, "Network Function Virtualisation (NFV) - Use Cases," Tech. Rep., 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[9] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[10] NFV ISG, "Network Function Virtualisation Infrastructure Architecture - Overview," Tech. Rep., 2014. [Online]. Available: http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/nfv-inf001v036-InfrastructureOverview.pdf

[11] ——, "Network Functions Virtualisation (NFV) - Virtualisation Requirements," Tech. Rep., 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf

[12] ——, "Network Function Virtualisation (NFV) - Resiliency Requirements," ETSI, Tech. Rep., 2014. [Online]. Available: http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/NFV-REL001v013-Resiliency%20Requirements.pdf

[13] VMWare Inc., "Vmware virtualization for desktop & server, application, public & hybrid clouds — united states." [Online]. Available: http://www.vmware.com/

[14] D. Lezcano, S. Hallyn, and S. Graber, "LXC - Linux Containers: Userspace tools for the Linux kernel containers." [Online]. Available: https://linuxcontainers.org/

[15] Openstack, "Openstack." [Online]. Available: http://www.openstack.org/

[16] T. Banzai, H. Koizumi, R. Kanbayashi, T. Imada, T. Hanawa, and M. Sato, "D-cloud: Design of a software testing environment for reliable distributed systems using cloud computing technology," in *Proc. Intl. Conf. CCGRID*, 2010, pp. 631–636.

[17] H. Fujita, Y. Matsuno, T. Hanawa, M. Sato, S. Kato, and Y. Ishikawa, "DS-Bench Toolset: Tools for dependability benchmarking with simulation and assurance," in *Proc. Intl. Conf. DSN*, 2012, pp. 1–8.

[18] X. Ju, L. Soares, K. G. Shin, K. D. Ryu, and D. Da Silva, "On fault resilience of OpenStack," in *Proc. SoCC*, 2013, pp. 1–16.

[19] P. Joshi, H. S. Gunawi, and K. Sen, "Prefail: A programmable tool for multiple-failure injection," in *Proc. Intl. Conf. OOPSLA*, 2011, pp. 171–188.

[20] Netflix, "Netflix Home Page." [Online]. Available: https://www.netflix.com

[21] A. Tseitlin, "The antifragile organization," *Commun. ACM*, vol. 56, no. 8, pp. 40–44, Aug. 2013.

[22] Amazon.com, Inc., "Amazon web services homepage." [Online]. Available: http://aws.amazon.com/

[23] Netflix, "The Chaos Monkey." [Online]. Available: https://github.com/Netflix/SimianArmy/wiki/Chaos-Monkey

[24] C. Pham, D. Chen, Z. Kalbarczyk, and R. K. Iyer, "CloudVal: A framework for validation of virtualization environment in cloud infrastructure," in *Proc. Intl. Conf. DSN*, 2011, pp. 189–196.

[25] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Proc. Linux Symp.*, vol. 1, 2007, pp. 225–230.

[26] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. SOSP*, 2003, pp. 164–177.

[27] RedHat. Virt-manager. http://virt-manager.et.redhat.com/.