

Experimental Evaluation of Resilience in an NFV-Based IP Multimedia Subsystem Using Fault Perturbation

Ayman Mohamed Bahaa Eldin

Electrical Power Engineering Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt.
bahaaeldinam@eng.cu.edu.eg

Hongkai Xiong

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
xiongho@tsinghua.edu.cn

Article Info

Elaris Computing Nexus
https://elarispublications.com/journals/ecn/ecn_home.html

Received 06 May 2026
Revised from 12 June 2026
Accepted 26 June 2026
Available online 08 July 2026
Published by Elaris Publications.

© The Author(s), 2026.

<https://doi.org/10.65148/ECN/2026010>

Corresponding author(s):

Hongkai Xiong, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
Email: xiongho@tsinghua.edu.cn

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract – Network Function Virtualization (NFV) has changed the face of telecom industry by replacing the hardware network services with software-based Virtual Network Functions (VNFs) that offer greater scalability and flexibility of services. On the other hand, virtualization problems with dependability are caused by infrastructure failure, software bugs and resource contention. An NFV based IP Multimedia Subsystem (IMS) is tested in an experiment and experience is gained in this paper to understand the robustness of the NFV-IMS via controlled fault injection. A quantitative assessment of service availability, latency, throughput and recovery time is complemented by a qualitative assessment of fault detection, isolation and recovery behavior. The results show how effective virtualization management techniques are at not only maintaining the continuity of service, but at also determining where the limits of resilience lie when operating in adverse conditions. The study provides concrete recommendations for robust telecom infrastructure design for NFV.

Keywords – Network Function Virtualization, IP Multimedia Subsystem, Fault Injection, Resilience Evaluation, Fault Tolerance, Virtual Network Functions, Service Availability.

I. INTRODUCTION

Network Function Virtualization (NFV) is the concept of replacing conventional hardware-based networking functions with software-oriented services called VNFs (virtualized network functions), which can run on general-purpose computer hardware [1], [2]. Unlike traditional network devices such as switches, firewalls, and routers, VNFs are deployable, scalable, and migratable dynamically across diverse hardware platforms, enabling the efficiency and agility of cloud computing models for network administration and service provisioning. Reliability and availability are two of the most important requirements in virtualized networks.

Due to the increased complexity and the number of different components, these systems are especially susceptible to unexpected outages due to hardware failures, software bugs, malfunctions and misconfigurations. Reliability is the capability of a model to perform its projected function under particular conditions for a specific duration of time without failure. Availability is the likelihood that a model is operational when needed, sometimes expressed as a percentage of time that the system is available over a specified period [3]. The NFV ISG has identified use cases, pre-requisites and frameworks that are going to be a benchmark for emerging NFV technologies. The telecommunications industry requires very high reliability standards, both from customers and from government regulations. The expected result of virtualized network functions is at least similar reliability to traditional networks, if not better. The expected NFVI specifications and framework, as described

by the ETSI in this section, include fault tolerance techniques for the next NFVIs. These design concepts will lead to NFVI fault tolerance solutions, which will consist of fault recovery, fault localization, and fault detection (see **Fig. 1**).

Hardware-accelerated virtualization is a basic concept for cloud-based applications and large clusters. Technological progress continuously generates regular hardware failures. These might cause serious dependability problems. Cinque et al. [4] introduce COSMOS, a fault injection framework aimed at the evaluation of hardware-assisted hypervisors. Target instrumentation is not required thanks to the use of nested virtualization, and COSMOS allows the evaluation of several hypervisors. They launched a large-scale fault injection operation to analyze hardware-based hypervisors, such as Jailhouse, Xen, and KVM. The findings indicate a large percentage of non-fail-stop operations, revealing considerable variances in the ability of hypervisors to record failures and mitigate the spread of faults by means of rapid recovery.

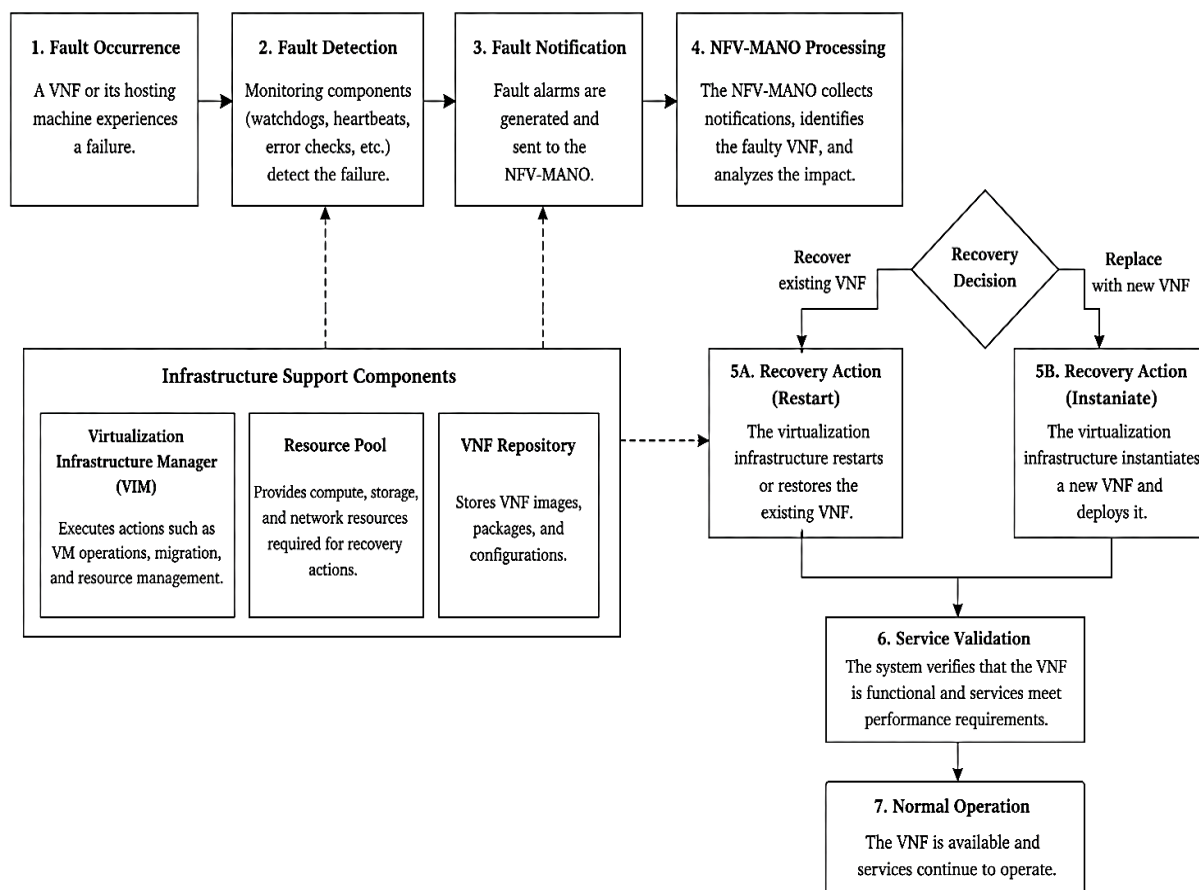


Fig 1. Overview of Fault Tolerance in NFV Infrastructures.

In [5], Ahmad and Andras undertake a study to investigate the effect of faults on the scalability and resilience of cloud-based software services. The research proposes an experimental paradigm based on Application-Level Fault Injection (ALFI) for studying the effects of application-level faults on the scalability resilience and performance of cloud-based software services. Existing work on assessing the scalability of cloud-based software services forms the basis for understanding scalability properties of software services and enables a full scalability analysis. Thokala and Gupta [6] conducted an experimental investigation on the EC2 cloud with a real cloud-based software service as an example of the architecture, considering the delay latency of software failures under two different setups and two demand conditions.

In relation to their study, Ahmad and Sedaghat [7] proposed BiGResi, a simulation-driven fault injection method to evaluate the software resilience at a granularity of instruction level. BiGResi mitigates the limits of existing random and instruction type-agnostic fault injection methods by measuring resilience in a fine-grained manner, visualizing both bit position and type of instruction. It also addresses the instruction set architecture (ISA). It raises the evaluation precision by revealing architecturally detectable faults. BiGResi adopts a timing-oriented fault injection approach with few changes in the target application, thus minimizing intrusiveness and keeping the performance almost native. We empirically evaluate the accuracy of BiGResi with many FI campaigns to target multiple benchmarks with different features.

However, despite these advances, very little research has been conducted to fully study the resilience of an NFV-based IMS architecture subjected to controlled fault injection and to relate the quantitative performance metrics with qualitative evaluation of operations. In this work, the robustness of an NFV-based IMS is discussed by investigating the effect of

infrastructure and virtualization faults on the availability, latency, throughput, recovery performance and the overall effectiveness of fault management [8] of the IMS.

The rest of this research is divided into the following sections. In Section II, the research methodology, research framework, experimental methods and data analysis techniques are outlined. The case study is presented in Section III where the system architecture, experimental scenario and procedure for resilience evaluation are described. Experimental results such as physical fault analysis, platform comparison and virtual fault analysis are included in Section IV. Finally, Section V provides a summary of the key findings, its implications for the resilience of IMS using NFV, and pointers for further research.

II. METHODOLOGY

Research Framework

The paper aims to propose an experimental research approach to measure resilience performance in NFV-based IP Multimedia Subsystem (IMS) environment in the presence of controlled fault scenarios. The method combines quantitative analysis of the performance with qualitative analysis of the systems' behavior to allow a full understanding of the dependability of virtualization platforms. This work is based on classical performance studies, which have been centered on numeric measures. Rather, it is service continuity, fault isolation, recovery effectiveness and operational manageability that are features that are more generic and make up resilience.

The proposed study framework is structured into five inter-related stages: (1) Preparation of the experimental environment (2) NFV based IMS deployment (3) Fault modelling & injection (4) Experimental environment Observation & Data gathering and (5) Resilience evaluation. The relationship between these stages is shown in Fig. 2 where the conceptual research design particularly developed for this study is mentioned along with the merging of experimental data with qualitative observations for evaluation. The experimental setup is designed to be a realistic, virtualized telecommunications platform in which the network services are running on shared computing, storage and network resources.

The system is instrumented with some known faults after deployment and the behavior of virtualization technologies in abnormal situations is observed. The data collected is then analyzed under normal and problem situations [9]. The qualitative part of the paradigm is used to gain a more complete picture of the system's behavior than the quantitative system is used to measure system performance. Fault events are identified rapidly, recovery procedures are triggered automatically, the service disruption is kept to a minimum and the time the service is out of operation and required to return to normal operation is analyzed. The observations demonstrate that there are times when challenges to resilience are not readily identified through performance measures. Operational effectiveness or technical performance is the proposed paradigm for what is meant by resilience. A virtualization platform with clear recovery processes that have reasonable latency and throughput would be more trustworthy than a platform that is more subject to manual recovery after failures [10].

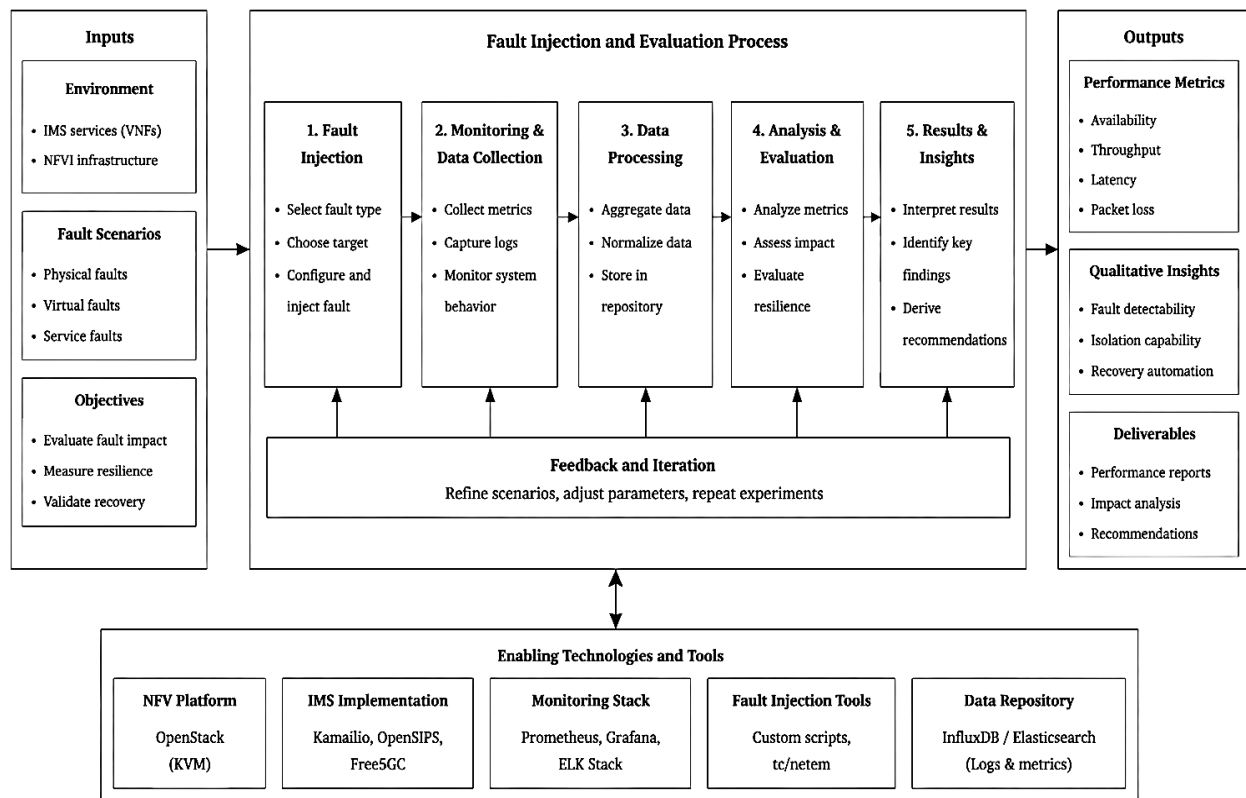


Fig 2. Proposed NFV Resilience Evaluation Technique.

Experimental Methodology

The experimental approach describes how to deploy, run and evaluate the experimental NFV-based IMS environment. The methodology adopted is the controlled experiment approach; that is, the same operational conditions are maintained and various failure conditions are introduced to study the behavior of the system. The first step of the experiment is the emulation setup of the virtualization infrastructure and deployment of IMS-related virtual network functions (VNFs). Once the deployment is successful, we generate the workload to bring about regular service conditions. Once the system stabilizes, we introduce defects at various levels of infrastructure. Fig. 3 depicts the whole experimental sequence, which represents the proposed process designed for the present research.

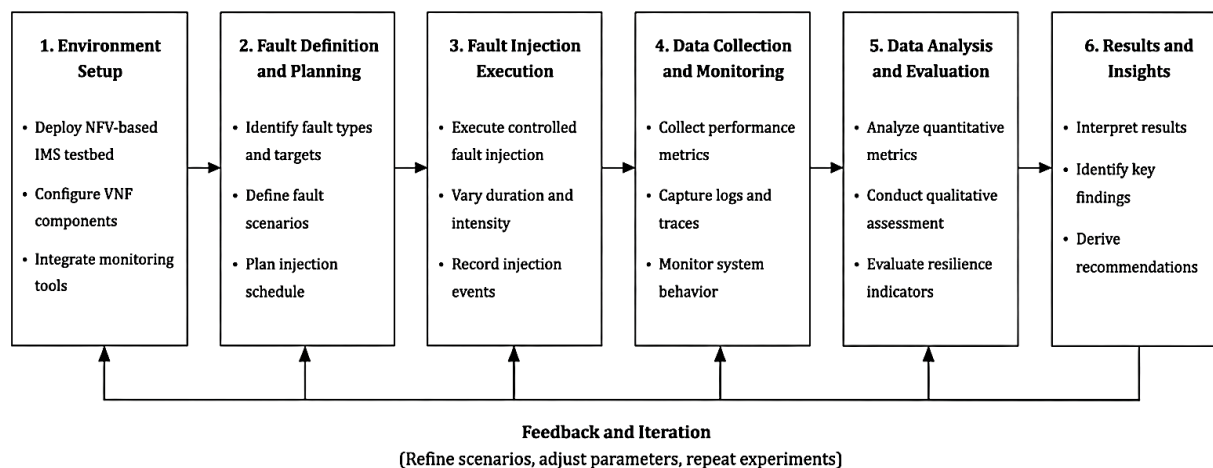


Fig 3. Experimental Approach Flow.

Resources (physical and virtual) may become corrupted. The fault injection model failures at the physical level are failures of fundamental infrastructure components like CPU resources, memory availability, storage operations and network connection [11]. Virtual level failures are failures that happen on virtual machines, containers or software network operations. These scenarios are used to evaluate virtualization technologies for maintaining services in various failure scenarios. The availability, latency, throughput and recovery length differences are calculated based on performance measurements during the research. However, the figures alone do not fully capture the consequences of failures. Thus, qualitative observations are also included by assessing system responses, recovery behavior, monitoring information and operational hurdles faced in fault events.

In the qualitative evaluation, the understanding of the practical consequences of failure handling is assessed. This involves determining whether the system provides adequate information on the causes of failures, whether recovery operations are predictable, and whether failures are contained to the components that are involved. It is crucial to make such discoveries because real telecommunications environments must be capable of high performance but also have predictable and controlled approaches to failure recovery. The methodology proposed uses controlled fault injection and performance measurement and qualitative assessment to provide an in-depth study of NFV resilience.

Data Analysis

Data analysis process is used in the comparison of the results obtained from the experimental works both in fault-free and fault-affected. The aim is to investigate the influence of different failure scenarios on the service performance and to identify parameters defining the system resilience. The quantitative analysis is the analysis of changes in the availability, throughput, latency and recovery times of the service. The degradation of performance is computed for each type of fault, each degree of degradation, by comparing the baseline measurements made in normal operation with measurements made during fault conditions [12]. Qualitative analysis is performed, as well as quantitative analysis, to attempt to make operational sense of the measured data. When issues are encountered, the system logs and the performance of the platform are examined to evaluate the performance of the virtualization solutions used. This study assesses the effectiveness of the fault detection techniques, fault isolation capability and degree of automation in carrying out the recovery procedure. The QQI leads to a deeper understanding of NFV resiliency. Performance measures indicate the extent of degradation of the service and qualitative evaluation indicates how the differences in the manner in which the faults are treated and operational reliability is achieved.

III. CASE STUDY

System Architecture

The proposed approach has been proved in an IP Multimedia Subsystem (IMS) environment, with a virtualized approach based on NFV. The setup for the experimental environment was designed to mimic a real deployment of a

telecommunications network, with network services realized as VNFs running on common computer resources. This design shows the evolution from proprietary hardware deployment to the software-defined network architecture used by service providers in today's world.

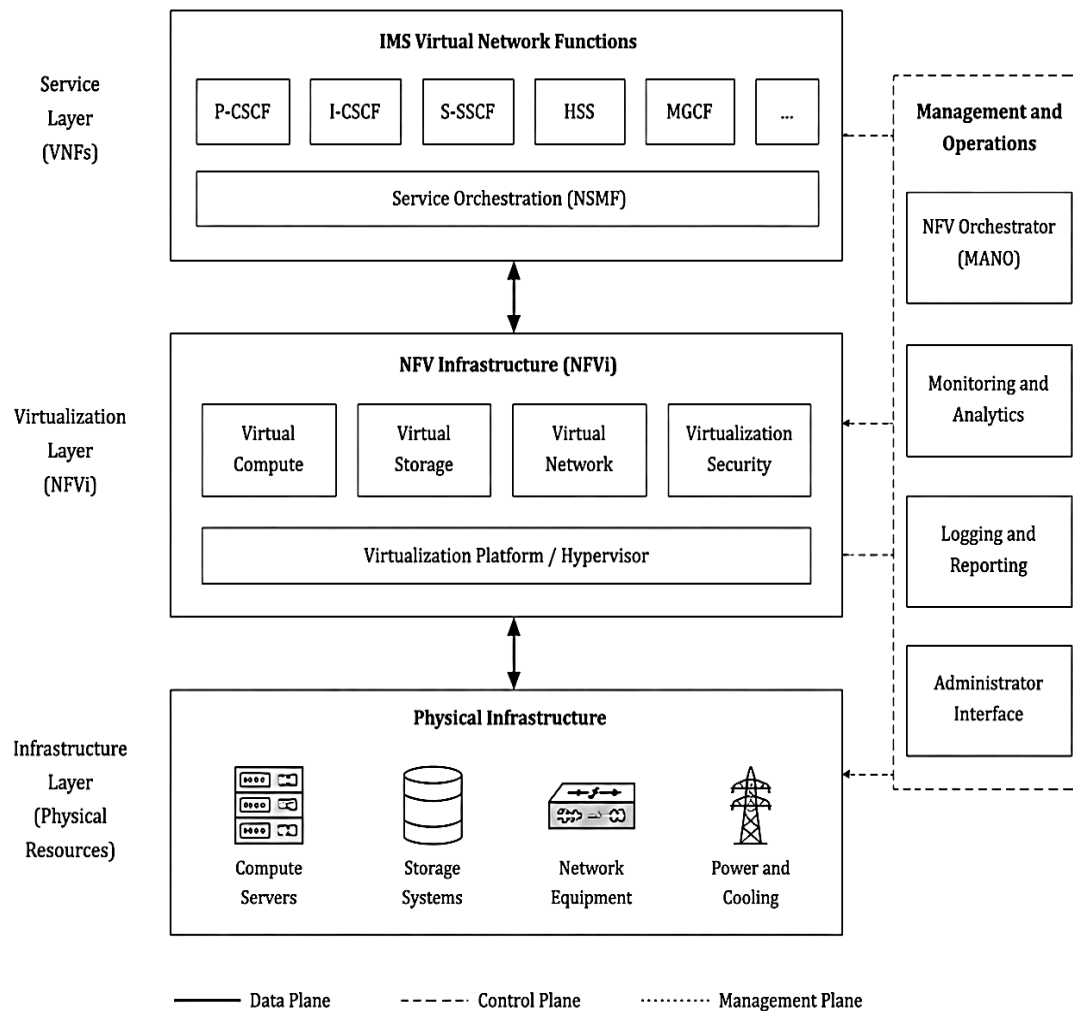


Fig 4. Proposed NFV-Based IMS Architecture for The Case Study.

The physical infrastructure layer offered computing, storage and networking resources; the virtualization layer was responsible for abstraction of resources and orchestration of VNFs; and the IMS service layer was responsible for communication services. The resource allocation, workload execution and fault recovery were managed using the virtualization platform and the ability to evaluate service continuity given varying infrastructure conditions. The whole experimental architecture used for the case study is shown in **Fig. 4**.

The selected environment was sufficiently robust to provide the basis for exploring the issue of resilience, with some VNFs sharing the same physical resources but being separated logically. This type of deployment mimics what is actually happening in the field, where multiple services could be affected at the same time if there is insufficient isolation and recovery methods if there are infrastructure failures, resource congestion or software anomalies. System performance and operational resilience were thus benchmarked in a representative deployment scenario, and the case study was therefore able to provide an assessment.

Experimental Scenario

We studied the resiliency of the deployed environment using a set of controlled fault injection experiments aimed at re-creating scenarios frequently observed in virtualized telecommunication systems. Each trial started with communication workloads typical of normal service conditions, and the IMS platform started up and was allowed to reach a stable state before the start of each trial. Baseline measurements were also taken for future reference.

Faults were injected independently into the infrastructure and virtualization layers to test their effect on service availability and recovery behavior. Infrastructure-level disruptions caused computing, memory, storage and network resources to be disrupted. Virtual machines, virtual network interfaces and software-based network operations were affected

by virtualization level failures. By introducing the fault sources one by one, it was possible to evaluate the contribution of each failure class before exploring more complex interactions among multiple fault sources.

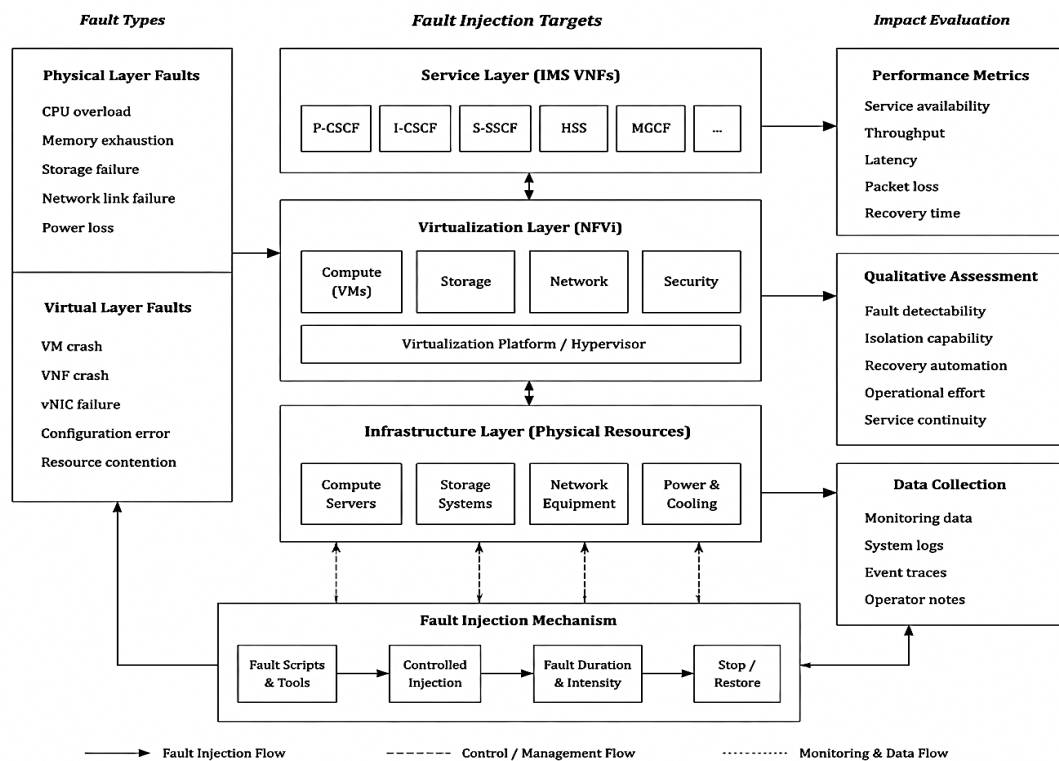


Fig 5. Scenario Model for Evaluation and Fault Injection.

Fig. 5 also depicts the experimental model used for the fault injection campaign. The model understands the relationship between the fault sources, affected components, monitoring activities and the consequent resilience evaluation. The behavior of the system was monitored continuously in all experiments to determine the service degradation caused by the disturbances added to the system and to check if the system recovered from each disturbance and how. This case study sets up repeatable experimental environments that enable evaluating the robustness of NFV-based IMS deployments by mimicking operational failures in a controlled environment.

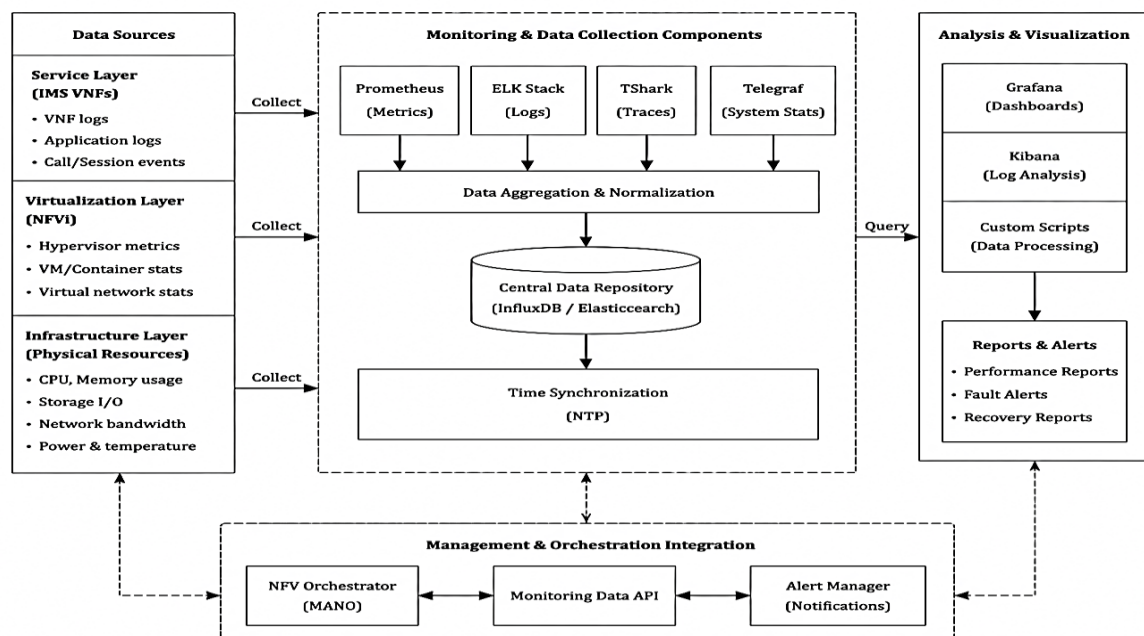


Fig 6. Monitoring and Data Collection Structure.

Experimental Workflow

The same sequence of trials was carried out for the various experiments, so as to ensure that the trials are consistent and repeatable for all fault conditions. IMS environment was successfully deployed in virtualized environment. Then workloads were injected for the services and the service's pre-defined faults were injected and workloads were allowed to stabilize. The system behavior was recorded during the experiment (before, during and after the fault incidence) by enabling performance monitoring. Quantitative information regarding service availability, latency, throughput and packet loss/recovery time was gathered by the monitoring platform.

Qualitative information was used to complement these metrics, which was collected from system logs, virtualization events and operational information. The effectiveness of the failure detection system, transparency of the recovery procedure and amount of work required to recover the provision of the normal service were given special attention. All of them were repeated under the same operating conditions in the experimental cases, to ensure the repeatability of the results. The observation made was compared to find out whether the behavior of the systems was uniform and the parts of the systems that were resilient to the multitude of failure types. The experimental campaign monitoring and data collection system is presented in **Fig. 6**. It illustrates the relationship between the monitored infrastructure, the data acquisition elements, the data repository and the resilience evaluation process.

Quantitative and qualitative measurements were slowly integrated to obtain a consolidated NFV resilience measurement. The numerical data would be utilized to provide information on impact on operating while the qualitative data will provide information on effectiveness of recovery mechanisms and practical issues in restoration during service. The integrated evaluation method allowed the assessment of the potential of virtualized IMS environments on providing reliable services under challenging operating scenarios.

IV. EXPERIMENTAL FINDINGS

In this section, we derived the metrics specified by our reliability standard from experimental data on performance and failures. To evaluate the decline of the IMS performance, we measured the throughput and latency of the IMS NFV model under faultless circumstances and compared these values with those in faulty cases as a baseline. Additionally, we propose that services should not be inaccessible for more than 30 seconds as a baseline for the evaluation of IMS unavailability. This decision is a positive limit of the network service unreachability, since the cost of key services may be even more constrained. However, as we will show with our study, it is still a very challenging task to achieve this objective using virtualized solutions.

Physical Fault Analysis

The inaccessibility of virtual functions, SIP REGISTERs (latency) and INVITEs & SIP REGISTERs (throughput) obtained from the fault perturbation trials are shown in **Fig. 7**, **Fig. 8** and **Fig. 9** for the Linux containers (Linux/Docker) and VMware platform (ESXi/vSphere) experimental setups. The graphs on the x-axis represent the different types of faults of (§ III-C. The trials demonstrated the great impact of the physical flaws on the inaccessibility of the NFV model. The degree of inaccessibility depends on the types of faults and the variations between Linux containers and VMware platform test setups. The NFV model was capable of resolving the challenges autonomously in all but two cases.

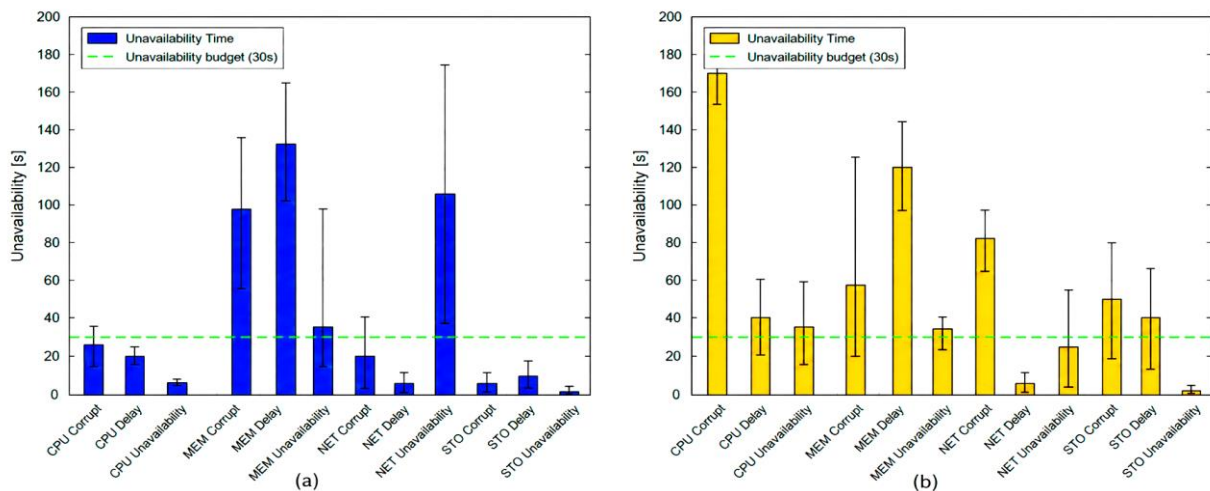


Fig 7. VNF Inaccessibility During Fault Perturbation (A) VMware VMWare Platform (B) Linux Containers.

Our analysis shows that introducing storage and network anomalies into the VMware platform environment stops the recovery procedure from completing, as the effects on fault management systems are long-lasting and cause a significant drop in availability. In such a scenario, the NFV model may require human intervention to manually reestablish the faulty copies. For the remaining fault categories, the system experiences a maximum of 100 seconds of downtime, while in the best scenarios, the virtual function downtime is maintained below the 30-second benchmark. More precisely, in the VMware

platform world, the impact of storage latency problems is negligible. This behavior can be defined based on the idea that virtual discs for the VMs reside on a commonly accessible iSCSI storage volume, which puts a small I/O workload on the local disc and hides latency issues quite well. CPU latency and down faults affect the Linux containers and VMware platform test setups to varying degrees.

While Linux-based containers are projected to reboot faster (since they are lighter weight than VMs), the amount of down time for these containers is greater than for VMware platform. In this situation, a long timeout interval largely impacts the recovery time by delaying Docker's definition of a node's failure. The discrepancy is worse with CPU computation faults. Within VMware ESXi, a failure of the kernel inside the hypervisor causes the hypervisor to crash and the workloads to be relocated. Conversely, with Linux containers, computation errors did not stop the host, but otherwise executed to the fullest extent possible, although tasks were stopped in kernel-level and internal system issues as evidenced by the log lines logged by the kernel. Nonetheless, the IMS queries on the defective hosts failed, and this fact prevents container transfer on the functional host.

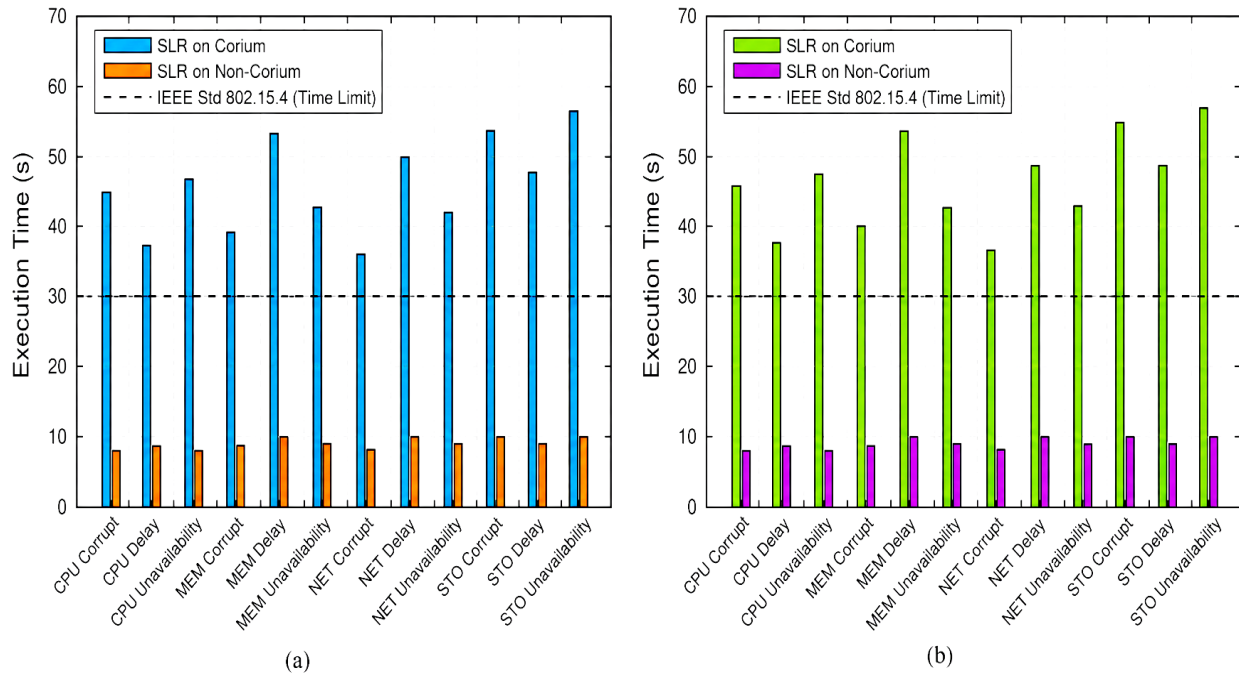


Fig 8. Virtual Function Output with Fault Perturbation (a) VMware VMware Platform (B) Linux Containers.

Platform Comparison

We observed another scenario of improper fault management when integrating memory concerns. The Linux containers environment is not as accessible as VMware platform, particularly when considering memory loss issues affecting the memory control mechanism as it relates to assignments. In this scenario, the NFVI has to safeguard the virtual function and provide sufficient resources for their operation, regardless of the interruption caused by the resource-consuming procedure [13]. Nevertheless, our results show that the process can impair the virtual functions' performance. The NFVI slowly recovers to its former state following the injection. The ESXi virtual machine monitor is more immune to these failures since it reserves memory for the virtual machines that the resource hog cannot steal. Linux containers offer less isolation between containerized environments since it uses more rudimentary memory control mechanisms which may result in delay due to excessive swapping and thrashing. In addition, the OS kernel enables the memory pressure handler in case of unavailability errors that leads to the virtual functions restart.

Network and storage failures have different impact on two NFV configurations as a result of the different architectures. VMware platform is highly intricate setup with more congestion over the system (like checking datastore liveness and locking, managing heartbeats, etc.) whereas Linux containers is a highly streamlined technology. In addition, the implementations (and thus the vulnerabilities) of TCP/IP differ between Linux and ESXi, and the entries vary greatly in their response to system anomalies (such as packets that were not processed, SYN flood alerts).

This is the same for the storage subsystem. The monitor is unable to create new threads or do I/O on the VMFS file subsystem due to a computation problem. For Linux containers, the virtual function inaccessibility is mitigated. It remounts the filesystem in read-only mode in the event of a failure of some application processes to prevent the fault from causing additional damage [14]. However, Linux containers become less accessible when there is delay/non-availability of storage, because of more I/O congestion on local storage. The efficiency measurements (see Fig. 8 and Fig. 9) in terms of throughput

and latency show a similar trend with virtual function unavailability, severe failures like storage and network calculations, and memory loss errors have a significant impact on efficiency.

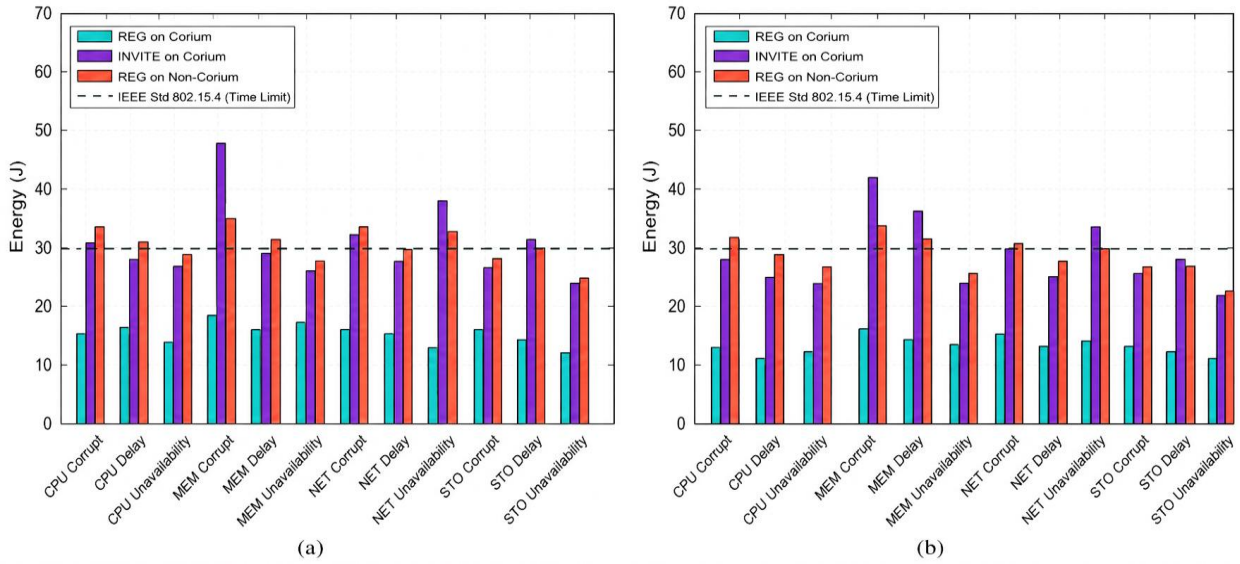


Fig 9. Virtual Function Latency Under Fault Perturbation (a) VMware Platform (b) Linux Containers.

Additionally, there are performance variations between Linux containers and VMware platform. CPU and memory failures often work better in VMware platform. This phenomenon stems from the capability of hypervisor-oriented virtualization to achieve better separation between virtualized services and machines. However, Docker provides reduced latency and increased throughput when system failures occur because its system design is more vigorous and capable of promptly restoring ideal service quality after they are introduced. For storage problems, VMware platform is highly tolerant to inaccessibility or storage latency but is prone to depletion in cases of corruption.

Virtual Fault Analysis

This section evaluates performance and downtime for defects in the NFVI virtual layer. **Fig. 10** depicts unavailability from fault perturbation investigations on Sprout and Homestead virtual functions in Linux containers and VMware platform environments. For brevity, we do not show the complete graphs for both virtual function latency and throughput but only the most relevant insights in the study. On VMware platform, it might cause more severe issues if there is a paucity of virtual CPUs. Linux containers can swiftly recover from loss of a virtual CPU (i.e. failure of the VM/container where the virtual function resides). In case of crash, VMware HA and Docker detect the termination of the virtual function (represented by a non-zero exit status in Linux) and swiftly recover the virtual function by automatically restarting it [15].

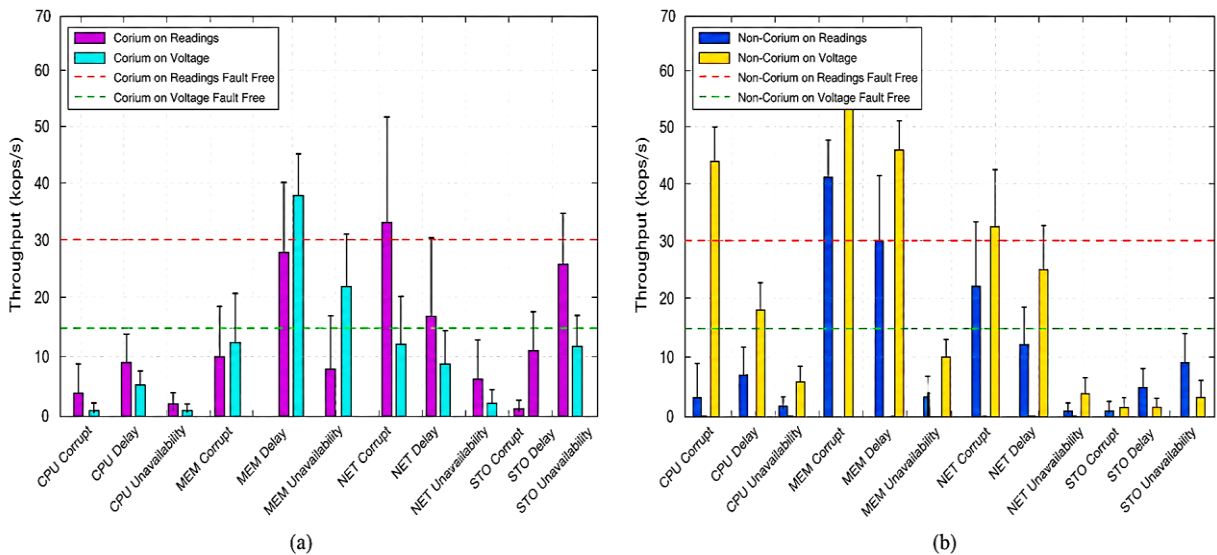


Fig 10. Virtual Function Unavailability on Fault Perturbation. (a) VMware Platform (b) Linux Containers.

A container requires less time to reinitialize than a virtual machine. It means restarting an operation instead of reconstituting all the hardware concepts simulated by the virtual machine monitor and restarting an entire guest operating system. However, for virtualized CPU and memory calculation, the variables are different. In Linux containers corruption disrupts the deployment of application procedures because the error modifies the inner condition of the virtual function tool, but the operating system does not trigger the cessation of the operations. In that sense, there is no reboot. In the VMware platform, the issue is perceived relatively early, as the ESXi hypervisor identifies the corruption and restarts the VM correctly. Problems with memory are more prevalent in the Linux containers testing environment, similar to the issues found in the physical network layer.

In VMware platform, if a virtual machine is pressuring the memory control system, the virtual machine may offer more isolation between virtual machines compared to Linux containers, due to allocated memory and a more effective memory allocation method. However, a non-functioning or overloaded virtual function in Linux containers configurations could propagate the performance reduction to additional virtual functions. This behavior is noticed in the evaluation of overall latency and throughput of the NFV model. Nonetheless, note that the VMware platform is still suffering from memory and CPU congestion because of an overloaded and malfunctioning virtual functions, since in some circumstances the inaccessibility of the virtual function is beyond the reference threshold. The virtual network fault perturbation suggests the Homestead virtual function necessity to provide IMS services.

This virtual function has deficiencies causing a substantial duration of inaccessibility. After reviewing the infrastructure logs, we concluded that the service failure was caused by a problem in the TCP communications (i.e., during the TCP window standardization). System latency issues affect Linux containers and VMware platform differently as a result of the differing internet protocol architectures. ESXi shows a higher level of network connectivity loss due to packet processing difficulties. Network outage concerns have a modest effect on downtime (approximately 3 seconds) as opposed to network corruption issues, as service performance is almost instantaneously resumed as the injector begins to drop packets. These issues are also applicable for the trials on sprout-based computing unit, except that Docker is far less vulnerable to system disturbances [16].

In this context, there is no difference between the Linux containers and VMware platform context in terms of unavailability and computation (for brevity, this difference is not shown in **Fig. 10**); however, given the context of computation, virtual functions have higher latencies in the Linux containers context (not shown for brevity). That's not the case with physical computation failures. The Linux operating systems running on the ESXi virtual machines were able to absorb the failure by reattaching the virtual disc to them in a non-writable state. The Docker case was when the Docker operating system was responsible for managing the panel and it did not reattach the virtual disc. A virtualized disc for ESXi provides additional strain to the virtual function, which is substantially influenced by latency difficulties in the storage system, since this disc contains the complete VM visualization and the root file system [17]. With Linux containers nevertheless, the virtualized disc will only have the partition which contains the Cassandra dataset.

Ultimately, the lack of storage availability issues does not affect either VMware platform or Linux containers. Evidence from such trials reveals that the NoSQL datastore on the afflicted virtualized node logs an I/O issue and shuts down to reduce further deterioration of the service, while the NoSQL datastore on other servers delivers the service requests.

V. CONCLUSION

This paper presents a controlled experiment to assess the robustness of an NFV-based IP Multimedia Subsystem (IMS) when facing infrastructure, virtualization and NFV instance faults. The goal is to examine the impact of failures in the infrastructure and virtualization layer on the service. Results show the impact of faults on the availability of service, latency, throughput and recovery characteristics and the requirement for robust fault detection and recovery mechanisms in virtualized environment. The system reliability was analyzed quantitatively and qualitatively for the operations, which provided a more profound understanding of system reliability than traditional performance indicators. The results show a true picture of the strength of NFV systems and in turn the need for positive fault management mechanisms for better continuity of services. As part of future research, we will investigate the adaptive recovery strategies and failure prediction based on machine learning to increase the resilience of large, virtualized telecommunication networks.

CRedit Author Statement

The author reviewed the results and approved the final version of the manuscript.

Conceptualization: Ayman Mohamed Bahaa Eldin and Hongkai Xiong; **Methodology:** Ayman Mohamed Bahaa Eldin and Hongkai Xiong; **Writing- Original Draft Preparation:** Ayman Mohamed Bahaa Eldin and Hongkai Xiong; **Investigation:** Ayman Mohamed Bahaa Eldin; **Supervision:** Hongkai Xiong; **Writing- Reviewing and Editing:** Ayman Mohamed Bahaa; All authors reviewed the results and approved the final version of the manuscript.

Data Availability

The datasets generated during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interests

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Funding

No funding was received for conducting this research.

Competing Interests

The authors declare no competing interests.

References

- [1]. C. Tipantuna and P. Yanchapaxi, "Network functions virtualization: An overview and open-source projects," in Proc. 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), pp. 1–6, Oct. 2017, doi: 10.1109/ETCM.2017.8247541.
- [2]. P. Shantharama, A. S. Thyagaturu, and M. Reisslein, "Hardware-Accelerated Platforms and Infrastructures for Network Functions: A survey of enabling technologies and research studies," IEEE Access, vol. 8, pp. 132021–132085, Jan. 2020, doi: 10.1109/ACCESS.2020.3008250.
- [3]. D. I. Heimann, N. Mittal, and K. S. Trivedi, "Availability and reliability modeling for computer systems," in Advances in Computers, 1990, pp. 175–233, doi: 10.1016/S0065-2458(08)60154-0.
- [4]. M. Cinque, D. Cotroneo, G. De Rosa, L. De Simone, and G. Farina, "COSMOS: A fault injection framework to assess hardware-assisted hypervisors," IEEE Trans. Dependable Secure Comput., vol. 22, no. 6, pp. 7448–7462, Aug. 2025, doi: 10.1109/TDSC.2025.3597103.
- [5]. A.-S. Ahmad and P. Andras, "Scalability resilience framework using application-level fault injection for cloud-based software services," J. Cloud Comput., vol. 11, no. 1, Jan. 2022, doi: 10.1186/s13677-021-00277-z.
- [6]. S. Thokala and S. Gupta, "Integrating cloud infrastructure for scalable web applications: Insights from AWS, EC2, and S3," in Proc. 2025 Int. Conf. Advancements Smart, Secure Intell. Comput. (ASSIC), pp. 1–6, May 2025, doi: 10.1109/ASSIC64892.2025.11158377.
- [7]. H. A.-H. Ahmad and Y. Sedaghat, "BiGResi: Robust bit-level fault injection framework for assessing intrinsic software resilience against soft errors," Comput. Electr. Eng., vol. 116, p. 109222, Apr. 2024, doi: 10.1016/j.compeleceng.2024.109222.
- [8]. D. Cotroneo, R. Natella, and S. Rosiello, "A fault correlation approach to detect performance anomalies in virtual network function chains," in Proc. 2017 IEEE 28th Int. Symp. Software Reliability Engineering (ISSRE), pp. 90–100, Oct. 2017, doi: 10.1109/ISSRE.2017.12.
- [9]. M. Le and Y. Tamir, "Fault injection in virtualized systems—Challenges and applications," IEEE Trans. Dependable Secure Comput., vol. 12, no. 3, pp. 284–297, Jul. 2014, doi: 10.1109/TDSC.2014.2334300.
- [10]. Z. Zhang et al., "UN-IOV: Achieving bare-metal level I/O virtualization performance for cloud usage with migratability, scalability and transparency," IEEE Trans. Comput., vol. 73, no. 7, pp. 1655–1668, Mar. 2024, doi: 10.1109/TC.2024.3375589.
- [11]. Y. B. Bekele, D. B. Limbrick, and J. C. Kelly, "A survey of QEMU-based fault injection tools & techniques for emulating physical faults," IEEE Access, vol. 11, pp. 62662–62673, Jan. 2023, doi: 10.1109/ACCESS.2023.3287503.
- [12]. F. Pelella, L. Viscito, and A. W. Mauro, "Soft faults in residential heat pumps: Possibility of evaluation via on-field measurements and related degradation of performance," Energy Convers. Manage., vol. 260, p. 115646, Apr. 2022, doi: 10.1016/j.enconman.2022.115646.
- [13]. J. Gonzalez, G. Nencioni, A. Kaminski, B. E. Helvik, and P. E. Heegaard, "Dependability of the NFV orchestrator: State of the art and research challenges," IEEE Commun. Surveys Tuts., vol. 20, no. 4, pp. 3307–3329, Jan. 2018, doi: 10.1109/COMST.2018.2830648.
- [14]. D. Cotroneo, L. De Simone, and R. Natella, "NFV-Bench: A dependability benchmark for network function virtualization systems," IEEE Trans. Netw. Service Manag., vol. 14, no. 4, pp. 934–948, Jul. 2017, doi: 10.1109/TNSM.2017.2733042.
- [15]. D. Cotroneo, L. De Simone, and R. Natella, "ThorFI: A novel approach for network fault injection as a service," J. Netw. Comput. Appl., vol. 201, p. 103334, Feb. 2022, doi: 10.1016/j.jnca.2022.103334.
- [16]. M. Di Mauro, G. Galatro, F. Postiglione, and M. Tambasco, "Performability of network service chains: Stochastic modeling and assessment of softwarized IP multimedia subsystem," IEEE Trans. Dependable Secure Comput., vol. 19, no. 5, pp. 3071–3086, May 2021, doi: 10.1109/TDSC.2021.3082626.
- [17]. S. Thyagaturu, P. Shantharama, A. Nasrallah, and M. Reisslein, "Operating systems and hypervisors for network functions: A survey of enabling technologies and research studies," IEEE Access, vol. 10, pp. 79825–79873, Jan. 2022, doi: 10.1109/ACCESS.2022.3194913.

Publisher's note: The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. The content is solely the responsibility of the authors and does not necessarily reflect the views of the publisher.

ISSN (Online): 3105-9082