Towards a Novel Model for Availability as a Service

José Flauzino
Elias P. Duarte Jr.
{jwvflauzino,elias}@inf.ufpr.br
Dept. Informatics, Federal University of Paraná
Curitiba, Brazil

ABSTRACT

Highly Available systems often require sophisticated, expensive techniques. However, end-users may not have enough resources on their premises, or the knowledge to make their systems reach that goal. It may be fully impractical to expect an end-user to easily transform a regular application into a highly available one. In this work, we propose a novel perspective on Availability as a Service (AaaS). The proposed model has the potential to offer different levels of availability in a simplified fashion. AaaS has been often used to describe clouds that offer highly available services. Our proposal, on the other hand, is for the environment itself to provide, on demand, the functionality required for end-user systems to achieve specific levels of availability. We also envision that AaaS can go beyond the cloud. This includes the case in which AaaS is provided in a scenario we call Edge-Core-Cloud Continuum, where the network core plays a prominent role through NFV-COIN (Network Functions Virtualization - COmputing In the Network).

CCS CONCEPTS

• **Computer systems organization** → **Availability**; • **Networks** → *Cloud Computing*; *In-network processing*.

KEYWORDS

Availability as a Service, AaaS, Cloud Computing, NFV-COIN

ACM Reference Format:
José Flauzino and Elias P. Duarte Jr.. 2023. Towards a Novel Model for Availability as a Service. In *Proceedings of Latin-American Symposium on Dependable Computing (LADC'23).* ACM, La Paz, Bolívia, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Computer systems are subject to failure. Availability refers to the readiness for the correct operation of a system, which reflects the probability that the system is ready to provide the service correctly when its users need it [2]. As computing systems have come to play critical roles for organizations and individuals, availability has become a crucial attribute. There is a pressing demand for highly available systems that are able to continue operating properly despite the failure of some of their components [19]. The demand may be very high, for example, in the context of telecommunications, the availability requirements are on the order of 99.999% - at most just over 5 minutes of service interruption per year.

Designing, implementing, and maintaining highly available systems are not trivial tasks. Those systems often require sophisticated and computationally expensive techniques to ensure the required availability levels. In particular, a highly available system typically has several mechanisms related to fault tolerance, including failure detection and recovery. Thus, from the perspective of developers and operators, building, deploying, and operating those systems requires deep knowledge of fault tolerance, which is not commonplace, even among IT professionals.

The last decades have been marked by the emergence of several "as a Service" (aaS) business models - and their respective technologies [12]. One of the main characteristics of these models is to provide some product (or functionality) to customers by abstracting various complexities and eliminating the need for customers to own the resources (hardware and/or software). Based on these models we have Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Database as a Service (DaaS), to name a few. In a slightly redundant way, the term Availability as a Service (AaaS) has been mainly employed by cloud computing platforms that offer infrastructure, software, storage, and other services on demand in a highly available fashion.

In this work, we propose a novel perspective for the AaaS model, which aims to transform the way end-user systems achieve different levels of availability. The key idea is that mechanisms related to system availability are provided as services for end-user systems to consume on demand and achieve a certain level of availability. Thus, instead of these systems needing to dispose of mechanisms that deal with ensuring their own availability, they can benefit from services offered by the environment itself. That is, in this AaaS perspective, end-user systems do not need to be concerned with issues such as failure detection, replica management, state recovery, and others, to ensure their own availability.

More specifically, we propose to offer resilience services that can be consumed on demand by end-user systems. As an example, we can consider a comparison with the IaaS model. In an IaaS platform, an end-user can choose the best computing service offering (such as flavors that have different capabilities of processing, storage, etc.) to meet their needs. Similarly, in the proposed AaaS model the end-user can choose from a set of resilience services that will offer different levels of availability for the user system to be deployed. It is also possible to provide services that are dynamically increased or decreased with resilience mechanisms to provide availability guarantees that meet SLAs (Service Level Agreements). In this line,

LADC'23, October 16–20, 2023, La Paz, Bolívia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

we propose an initial set of categories of resilience services that are composed by combining different mechanisms. Each category aims to provide distinct levels of availability for systems, with varying computational costs.

In addition, we envision that AaaS can even go beyond the cloud. First, to achieve the desired levels of availability, both user system components and resilience service mechanisms can be allocated at points other than the cloud, such as at the edge. However, we can also extend some responsibility for the availability of the end-user systems to the network by applying the concept of NFV-COIN (Network Functions Virtualization - COmputing in the Network) [43]. Hence, this converges into a scenario we call Edge-Core-Cloud Continuum, where the network core plays a prominent role.

The rest of the paper is organized as follows. In Section 2 we present the background, including an overview of the main involved concepts. Related work is discussed in Section 3. In Section 4 we propose the novel perspective for AaaS and introduce the key ideas for its provision at the Edge-Core-Cloud Continuum based on NFV-COIN. Finally, Section 5 concludes the paper and describes future works.

2 BACKGROUND

This section provides a brief introduction to the main concepts behind our proposal. First, we describe the central aspects of "aaS" business models. Next, we present the NFV paradigm and then an overview of the integration of paradigms that have come to converge in the NFV-COIN concept.

2.1 "aaS" Provision Models

"As a Service" (aaS) are business models in which some kind of functionality or resource is offered to the customer in a service form. In these models, instead of the customers needing to purchase and maintain specific hardware or software, they just consume (use) on demand the functionality/resource that is offered as a service by a third party. This has been widely and successfully applied in the context of cloud computing. Currently, several models are based on these principles, ranging from broader models such as IaaS, PaaS, and SaaS, to more specific ones such as STaaS (STorage as a Service), DaaS (Database as a Service), and others [12, 27].

Beyond the business model itself, the abstraction provided by offering resources and functionality as a service makes building systems at higher layers easier. Achieving feasible solutions requires defining architectures, technical strategies, and implementing systems (or platforms) to deliver the services properly.

2.2 Network Functions Virtualization

The NFV paradigm emerged as an alternative to hardware-based network middleboxes. NFV technology allows the implementation of network functions in software. The so-called Virtualized Network Functions (VNFs) can be executed on off-the-shelf hardware. Moreover, multiple VNFs can be chained together in predefined orders to compose complex network services called *Service Function Chains* (SFC) [7]. Multiple services can be further combined into complex service topologies [17].

NFV represents a huge leap in terms of flexibility, being based on software decoupled from hardware. The main advantages are in terms of the reduction of the cost to build and operate the network. Compare the acquisition of a dedicated hardware solution with downloading the function from an NFV marketplace [4]. The list of advantages also includes scalability, ease of migration, reduced time for deployment, reduced need for physical space and energy, ease to upgrade and remove network functions, and more flexibility in composing network services [20, 30].

The European Telecommunications Standards Institute (ETSI) has coordinated standardization efforts to define and allow the interoperability of NFV solutions. The NFV-MANO (NFV Management and Orchestration) reference architecture [32] defines a set of specifications related to the management and orchestration of virtualized network services, providing standardized communication interfaces and resource abstractions to support the execution of heterogeneous services [15]. Three particularly important blocks are *i*) the VNF Manager (VNFM), responsible for managing the lifecycle of VNFs; *ii*) the Virtualized Infrastructure Manager (VIM), charged with the management and orchestration of computing resources in the infrastructure; and *iii*) the NFV Orchestrator (NFVO), responsible for network service lifecycle management and resource orchestration (supported by the VIM).

2.3 Computing in the Network & NFV-COIN

In-network computing allows part of the computing traditionally executed as end-user applications on end-user devices and/or the cloud to be performed on network devices. Most of the current work on in-network computing is based on programmable hardware, such as ASICs (Application Specific Integrated Circuits) and FPGA (Field Programmable Gate Array). An increasingly large number of applications has been reported [24, 37]. Next, we give an overview of a few significant landmarks.

In [9, 10] the Paxos consensus algorithm was implemented on a switch using P4. Other in-network computing services implemented with programmable hardware include data storage systems [5] and a distributed data partition/aggregation application [33]. Caching is perhaps one of the most intuitive applications of in-network computing. Several works have described caching solutions in the context of key-value stores [23, 28, 37, 44]. Among those, [23] stands out in terms of performance, reducing the latency up to 40% and increasing the throughput 10x in comparison with a traditional alternative. Network security has also been addressed with in-network computing. For example, [16, 25, 29, 47] propose in-network computing approaches to mitigate DDoS attacks. In [31] network telemetry operations are scheduled on programmable switches to improve the scalability of network-wide telemetry with respect to dynamic traffic and query loads.

In the context of the edge-cloud continuum, the integration of network computing and network processing in a common framework has been called Computing in the Network (COIN) [46]. Recently, the concept of NFV-COIN was introduced in [43]. That work can be seen as the starting point for the integration of those two paradigms. Besides discussing the requirements for deploying NFV-based COIN services, a draft NFV-COIN architecture compliant with the NFV-MANO reference model is presented. The authors also present successful NFV-COIN use cases [39, 41, 42].

Table 1: Categories of resilience services and their mechanisms.

	Category	Failure Detector	Failure Recovery	Replica Manager		Checkpointer	State Base	Replica State Synchronizer	
				Passive	Active			Reactive	Active
Stateless	SL-1	✓	✓	-	-	-	-	-	-
	SL-2	✓	✓	✓	-	-	-	-	-
	SL-3	✓	✓	-	✓	-	-	-	-
Stateful	SF-1	✓	✓	-	-	✓	✓	✓	-
	SF-2	✓	✓	✓	-	✓	✓	✓	-
	SF-3	✓	✓	-	✓	✓	✓	-	✓

3 RELATED WORK

The acronyms AaaS or even HAaaS (High Availability as a Service) have been used with different meanings. For example, certain vendors employ these terms when offering software and/or hardware solutions in a service format that presents high availability properties [35, 36]. Such terms have also been used in the context of database and storage systems that apply sophisticated strategies to achieve a certain level of availability [22, 45].

In a slightly redundant way, AaaS has been also employed when referring to cloud computing platforms that offer infrastructure, software, storage, and other services on demand in a highly available fashion [6, 8]. However, actually, cloud computing platforms employ high availability functionalities to provide their services in a highly available manner [21]. In this work, on the other hand, we propose to define an approach to enable the provision of services that can be consumed on demand by systems aiming to achieve some desired level of availability.

As mentioned earlier, we envision that NFV-COIN has the potential to enable our view of AaaS on the Edge-Core-Cloud Continuum. This idea is also encouraged by recent works demonstrating promising results when applying NFV-COIN [39, 41, 42]. In particular, in [39] a failure detector based on NFV-COIN was proposed. The solution, called NFV-FD (NFV - Failure Detector) uses information obtained from SDN (Software-Defined Network) controllers to monitor processes and determine their states (*i.e.*, correct or suspected to have crashed). In [42] the network itself offers reliable and ordered broadcast services to end-user applications, which can be extended to provide a publish-subscribe in-network service [11].

4 A NOVEL PERSPECTIVE ON AAAS

In this section, we discuss a new perspective for AaaS, in which instead of end-user systems being responsible to guarantee the availability of their own systems and applications, they can benefit from the services offered by the environment itself. We employ the term *environment* to denote in a generic way the universe set U of which all systems and services are part. In practice, U has a boundary and can represent a domain, a federation of domains, or even a more specific system, such as a private cloud.

Consider an end-user system, or just *system*, that is managed by the end-user, and can be deployed on the user's premises or on another location, such as the cloud. A system consists of a set of components C, such that $C \neq \emptyset$. Thus, a system can be monolithic ($|C| = 1$), centralized, decentralized, or distributed. Each component can play different roles in the system. We also assume that a system can be composed of components that are stateless, stateful, or both (*i.e.*, part of the components are stateless, part are stateful).

Ensuring the fault tolerance of a "generic" system is a non-trivial task. Traditionally, dependable systems involve the integration of resilience mechanisms into the system so that it is able to (transparently) recover on its own from eventual failures. In this sense, the proposed alternative is that these mechanisms are provided in the form of services by the environment in which the system is running. Thus, consider M to be the set of all existing *resilience mechanisms* in an environment U. An *service* S is composed of a specific selection of mechanisms in M, *i.e.*, $S \subseteq M \subset U$.

For example, assume that M is composed of the following resilience mechanisms: failure detector; failure recovery, passive and active replica manager; checkpointer; state base; and a replica synchronizer for active and reactive mode (*i.e.*, synchronizes state between replicas periodically or only when it is necessary to react to a failure). From these, various combinations can be made to compose services that offer different levels of availability, with different computational costs. Considering each unique valid combination as a category, Table 1 presents possible categories of services to be composed from the set of mechanisms mentioned. Table 2 briefly describes each category.

An advantage of this approach is that, as long as the implementations of each mechanism are compatible with each other, a resilience service can be transformed from one category to another one just by adding or removing mechanisms. This offers greater flexibility for the provider. For example, since the mechanisms in Table 1 are indexed from 1 to 6, an SF-1 service would be composed of $S = \{M_1, M_2, M_4, M_5, M_6\}$, with M_6 set to be reactive. Thus, the service SF-1 could be transformed into SF-3 from an operation $S \cup \{M_3\}$ (with M_3 set as active), and the shifting of M_6 from reactive to the active mode.

Another benefit is that subsets of the components of the same system can take advantage of different services. This allows, for example, critical components of a system to adopt a more robust resilience service (such as SL-3 for stateless components, or SF-3 for stateful ones), while the remaining components are served by less financially and computationally costly services (like SL-1 and SF-1).

The proposed approach also enables the inclusion of optimization strategies in the composition of resilience services. For illustration, consider that each mechanism M_i supports managing up to λ system components. Even if there is a single instance of each mechanism of Table 1 in an environment U, it is possible to compose different

Table 2: Description of the resilience services categories.

	Cat.	Description
Stateless	SL-1	There is no replication. In case the system component crashes, another instance of the component is created from scratch.
	SL-2	There are one or more replicas *in standby mode*. In case the primary one crashes, another replica is activated to take over.
	SL-3	There are one or more *active* replicas, one of which is the primary. In case the primary replica crashes, another replica takes over immediately.
Stateful	SF-1	There is no replication. The system component state is persisted periodically. In case the component crashes, another instance is created and the most recent persisted state is applied to it.
	SF-2	There are one or more replicas *in standby mode*. The state of the primary is persisted periodically. In case the primary one crashes, another replica is activated with the latest persisted state (becoming the primary one).
	SF-3	There are one or more *active* replicas, one of which is the primary. The state of the primary is periodically persisted and synchronized to the other replicas. In case the primary one crashes, another one takes over already having the most recent persisted/synchronized state.

services through logical assignments - respecting the λ limitation of each mechanism. For example, the transformation of an SF-1 service to SF-3 exemplified above can be done just by assigning a set of components C to be managed by mechanisms M_1, M_2, M_4, M_5, and M_6, then also assigning them to M_3, provided that $|C| \leq \lambda \; \forall \{M_1, M_2, M_3, M_4, M_5, M_6\}$.

4.1 AaaS vs. IaaS: Differences and Similarities

Although the AaaS model that we propose is clearly different from the other related models mentioned earlier, the AaaS services described above does have some similarities the IaaS model, in particular in terms of functionalities. For example, in some of today's IaaS cloud platforms, it is common to have features such as self-healing and replication of virtual machines (VMs) or containers (with load balancing between replicas). However, typically the services provided by the IaaS model do not extend beyond that.

Thus, in general, IaaS platforms do not cover, for instance, cases where the process of the end-user system component has died (even with the VM running), the automatic saving and restoring of stateful components, among other relevant issues. This is entirely reasonable, as it is outside the scope of IaaS to address these issues. In this context, our AaaS proposal emerges as a complementary model to IaaS, bringing a vision focused on end-user systems. On the other hand, AaaS can also leverage IaaS just as models like PaaS and SaaS do.

4.2 On the Availability of Resilience Services

A central benefit of the proposed AaaS model is that much of the responsibility for the availability of the end-user systems is offloaded to the resilience services provided by the environment. We argue that it would allow end-user systems to be simpler and leaner, while still being highly available. However, an extra issue is that the resilience services themselves need to be highly available, as the outage of these services can directly affect the availability of end-user systems. Actually, the environment itself should be highly available, by including mechanisms such as fault-tolerant routing [14] or even the ability to select stable resources to run the resilience services [13].

Therefore, offering AaaS requires that the resilience mechanisms (such as those presented in Table 1) are designed and implemented to be highly available and self-reliant. In practice, each resilience mechanism can be a distributed fault-tolerant system embedded in the service provider's ecosystem (*e.g.*, a cloud platform) or even deployed on top of it. The maturity of today's service providers makes this a reasonable assumption.

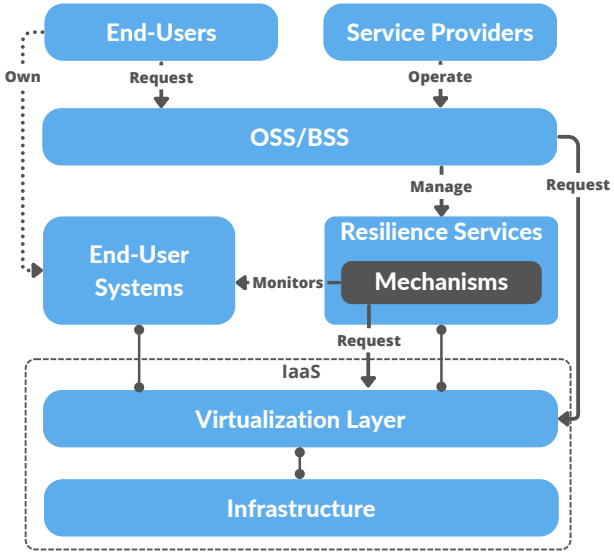

Figure 1: AaaS architecture.

4.3 Towards an AaaS Architecture

As a work in progress, we are designing an architecture for AaaS. Figure 1 illustrates the conceptual architecture that will serve as a starting point for defining a future comprehensive architecture for AaaS. In the architecture, AaaS providers can manage the resilience services catalog by operating the OSS/BSS (*i.e.*, the AaaS platform). Through OSS/BSS, end-users can also request on-demand resilience services for their systems. Both end-user systems and resilience services are deployed over a virtualization layer. Proper management

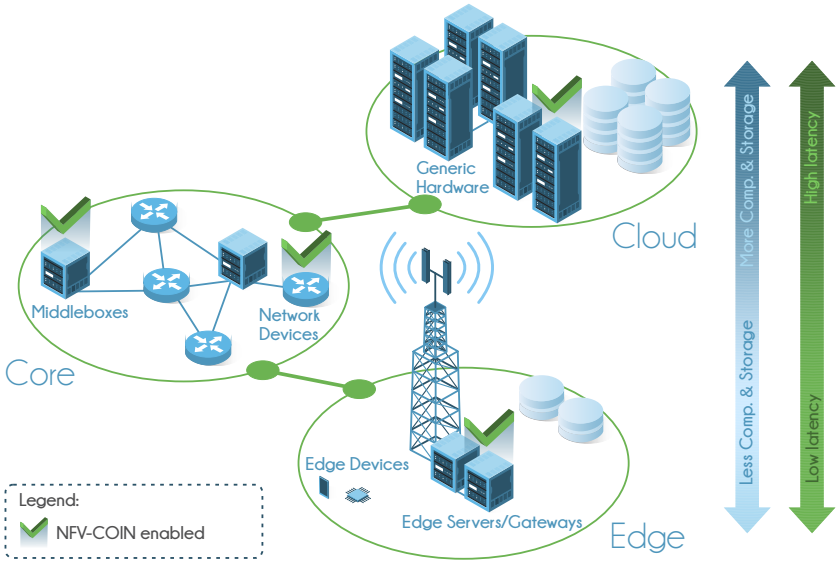

Figure 2: NFV-COIN in the Edge-Core-Cloud Continuum.

of services and systems can be achieved through pre-established communication between OSS/BSS and an underlying IaaS platform.

4.4 Expanding AaaS to the Edge-Core-Cloud Continuum

Cloud computing has been widely adopted around the globe to offer a wide range of on-demand services over the Internet. The multiple advantages of system outsourcing have led to a world-wide trend of organizations moving their data and applications to the cloud. However, requirements such as very low latency *and* high throughput; the need for location awareness; mobility support; privacy issues; among others, have fostered the search for new complementary alternatives to the cloud. In this context, *Edge Computing* has emerged [34] with the primary purpose to employ resources closer to end-users, reducing the negative effects of using resources over long distances without a sense of locality [1].

In recent years, the blending of the cloud and edge paradigms plus other key technologies has converged into the so-called Edge-Cloud Continuum [46]. In this scenario, systems can be deployed either on the cloud, the edge, or even distributed along the way, according to performance requirements and other issues [3].

We believe AaaS can also take advantage of the Edge-Cloud Continuum, by placing resilience services closer to the user (*e.g.*, on the *edge*) or farther away in locations with a higher concentration of computing resources, such as the *cloud*. We envision that NFV-COIN enables the construction of resilience services. In particular, NFV-COIN can allow an extension of the Edge-Cloud Continuum to include the core of the network. NFV-COIN services are by definition in-network services that can execute anywhere along a Edge-Core-Cloud Continuum. Basic resiliency services have already been implemented based on this technology. As mentioned earlier, an NFV-COIN-based failure detector [38, 39]can analyze network traffic to monitor processes (or system components); network caching based on NFV-COIN (similar to [23]) can optimize data transactions regarding checkpoints of system states; among many other possibilities.

As shown in Figure 2, both end-user systems and resilience services can be allocated along the Edge-Core-Cloud Continuum. Since NFV is already a reality in many Internet Service Providers (ISPs), NFV-COIN-based resilience services implemented as VNFs can be deployed in the network core with no change to the existing infrastructure.

5 CONCLUSION

In this Student Forum paper, we discussed a new perspective on AaaS. In the proposed draft AaaS model, resilience services composed of multiple mechanisms are provided on demand and can be employed to make end-user systems achieve certain levels of availability. We have defined multiple categories of services that offer different levels of availability by consuming varied computational resources. Some of the categories represent services that can support stateful system components and others only stateless ones. We also propose that the draft AaaS architecture can employ NFV-COIN technology and take advantage of the Edge-Core-Cloud Continuum.

As future work we envision that the proposed AaaS model can be applied to specific scenarios, such as the Internet-of-Things [18] or vehicular and sensor networks [26]. We also believe the architecture can be further extended to include other dependability services, such as those related to security [40]. We are currently working on the draft architecture. Several issues have to be clarified, including the relationships among the involved elements and also with other related architectures (such as the NFV-COIN architecture). Near future work also includes the implementation of a prototype as a Proof of Concept (PoC) that allows us to evaluate potential AaaS use cases, which includes its expansion to the Edge-Core-Cloud Continuum.

6 ACKNOWLEDGMENTS

This work has been partially supported by CAPES, Brazil - Finance Code 001; and the Brazilian National Council for Scientific and Technological Development (CNPq) - grant 308959/2020-5.

REFERENCES

[1] Ejaz Ahmed, Arif Ahmed, Ibrar Yaqoob, Junaid Shuja, Abdullah Gani, Muhammad Imran, and Muhammad Shoaib. 2017. Bringing computation closer toward the user network: Is edge computing the solution? *IEEE Communications Magazine* 55, 11 (2017), 138–144.

[2] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing* 1, 1 (2004), 11–33.

[3] Emmanuel Bertin, Noël Crespi, and Thomas Magedanz. 2021. *Shaping Future 6G Networks: Needs, Impacts, and Technologies.* John Wiley & Sons.

[4] Lucas Bondan, Muriel F Franco, Leonardo Marcuzzo, Giovanni Venancio, Ricardo L Santos, Ricardo J Pfitscher, Eder J Scheid, Burkhard Stiller, Filip De Turck, Elias P Duarte, et al. 2019. FENDE: marketplace-based distribution, execution, and life cycle management of VNFs. *IEEE Communications Magazine* 57, 1 (2019), 13–19.

[5] Pietro Bressana, Noa Zilberman, Dejan Vucinic, and Robert Soulé. 2020. Trading latency for compute in the network. In *Proceedings of the Workshop on Network Application Integration/CoDesign.* 35–40.

[6] Frederico Manuel Duarte Cerveira. 2021. *Evaluating and improving cloud computing dependability.* Ph. D. Dissertation. 00500:: Universidade de Coimbra.

[7] Margaret Chiosi, Don Clarke, Peter Willis, Andy Reid, James Feger, Michael Bugenhagen, Waqar Khan, Michael Fargano, Chunfeng Cui, Hui Deng, et al. 2012. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow world congress*, Vol. 48. sn, 1–16.

[8] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, and Andrew Warfield. 2008. Remus: High availability via asynchronous virtual machine replication. In *Proceedings of the 5th USENIX symposium on networked systems design and implementation.* San Francisco, 161–174.

[9] Huynh Tu Dang, Pietro Bressana, Han Wang, Ki Suh Lee, Noa Zilberman, Hakim Weatherspoon, Marco Canini, Fernando Pedone, and Robert Soulé. 2020. P4xos: Consensus as a network service. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1726–1738.

[10] Huynh Tu Dang, Marco Canini, Fernando Pedone, and Robert Soulé. 2016. Paxos made switch-y. *ACM SIGCOMM Computer Communication Review* 46, 2 (2016), 18–24.

[11] João Paulo de Araujo, Luciana Arantes, Elias P Duarte Jr, Luiz A Rodrigues, and Pierre Sens. 2019. VCube-PS: A causal broadcast topic-based publish/subscribe system. *J. Parallel and Distrib. Comput.* 125 (2019), 18–30.

[12] Yucong Duan, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud C Narendra, and Bo Hu. 2015. Everything as a service (XaaS) on the cloud: origins, current and future trends. In *2015 IEEE 8th International Conference on Cloud Computing.* IEEE, 621–628.

[13] Elias P Duarte, Thiago Garrett, Luis CE Bona, Renato Carmo, and Alexandre P Züge. 2010. Finding stable cliques of PlanetLab nodes. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN).* IEEE, 317–322.

[14] Elias Procópio Duarte Jr, Rogério Santini, and Jaime Cohen. 2004. Delivering packets during the routing convergence latency interval through highly connected detours. In *International Conference on Dependable Systems and Networks, 2004.* IEEE, 495–504.

[15] ETSI. 2015. *Network Functions Virtualisation (NFV); Infrastructure Overview.* Technical Report. European Telecommunications Standards Institute.

[16] Kurt Friday, Elie Kfoury, Elias Bou-Harb, and Jorge Crichigno. 2020. Towards a unified in-network DDoS detection and mitigation strategy. In *2020 6th IEEE Conference on Network Softwarization (NetSoft).* IEEE, 218–226.

[17] Vinicius Fulber-Garcia, Elias P Duarte Jr, Alexandre Huff, and Carlos RP dos Santos. 2020. Network service topology: Formalization, taxonomy and the custom specification model. *Computer Networks* 178 (2020), 107337.

[18] Thiago Garrett, Schahram Dustdar, Luis CE Bona, and Elias P Duarte. 2018. Traffic differentiation on internet of things. In *The 12th IEEE Symposium on Service-Oriented System Engineering (SOSE).* IEEE, 142–151.

[19] Jim Gray and Daniel P. Siewiorek. 1991. High-availability computer systems. *Computer* 24, 9 (1991), 39–48.

[20] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. 2015. Network function virtualization: Challenges and opportunities for innovations. *IEEE communications magazine* 53, 2 (2015), 90–97.

[21] Gustavo B Heimovski, Rogerio C Turchetti, Juliano A Wickboldt, Lisandro Z Granville, and Elias P Duarte Jr. 2020. FT-Aurora: A highly available IaaS cloud manager based on replication. *Computer Networks* 168 (2020), 107041.

[22] Chetan Jaiswal and Vijay Kumar. 2015. DBHAaaS: Database high availability as a service. In *11th SITIS.* IEEE, 725–732.

[23] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. 2017. Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles.* 121–136.

[24] Somayeh Kianpisheh and Tarik Taleb. 2022. A Survey on In-network Computing: Programmable Data Plane And Technology Specific Applications. *IEEE Communications Surveys & Tutorials* (2022).

[25] Peng Kuang, Ying Liu, and Lin He. 2020. P4DAD: Securing duplicate address detection using P4. In *ICC 2020-2020 IEEE International Conference on Communications (ICC).* IEEE, 1–7.

[26] Neeraj Kumar, Al-Sakib Khan Pathan, Elias P Duarte, and Riaz Ahmed Shaikh. 2015. Critical applications in vehicular ad hoc/sensor networks. *Telecommunication Systems* 58 (2015), 275–277.

[27] David S Linthicum. 2009. *Cloud computing and SOA convergence in your enterprise: a step-by-step guide.* Pearson Education.

[28] Zaoxing Liu, Zhihao Bai, Zhenming Liu, Xiaozhou Li, Changhoon Kim, Vladimir Braverman, Xin Jin, and Ion Stoica. 2019. DistCache: Provable Load Balancing for Large-Scale Storage Systems with Distributed Caching.. In *FAST*, Vol. 19. 143–157.

[29] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches.. In *USENIX Security Symposium.* 3829–3846.

[30] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2015. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials* 18, 1 (2015), 236–262.

[31] Chris Misa, Ramakrishnan Durairajan, Reza Rejaie, and Walter Willinger. 2021. Revisiting network telemetry in coin: A case for runtime programmability. *IEEE Network* 35, 5 (2021), 14–20.

[32] Jrgen Quittek, Prashant Bauskar, Tayeb BenMeriem, Andy Bennett, Michel Besson, et al. 2014. Network functions virtualisation (nfv)-management and orchestration. *ETSI NFV ISG, White Paper* (2014), 0733–8716.

[33] Amedeo Sapio, Ibrahim Abdelaziz, Abdulla Aldilaijan, Marco Canini, and Panos Kalnis. 2017. In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks.* 150–156.

[34] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.

[35] Mohamed Sohail, Emanuela Caramagna, and Sameh Gad. 2016. *High Availability as a Service (HAaaS).* Technical Report. Dell EMC.

[36] Abacus Solutions. [n. d.]. High Availability as a Service (HAaaS). https://www.abacusllc.com/high-availability-as-a-service-haaas

[37] Yuta Tokusashi, Hiroki Matsutani, and Noa Zilberman. 2018. Lake: the power of in-network computing. In *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig).* IEEE, 1–8.

[38] Rogério C Turchetti, Elias P Duarte, Luciana Arantes, and Pierre Sens. 2016. A QoS-configurable failure detection service for internet applications. *Journal of Internet Services and Applications* 7, 1 (2016), 1–14.

[39] Rogerio C Turchetti and Elias P Duarte Jr. 2017. NFV-FD: Implementation of a failure detector using network virtualization technology. *International Journal of Network Management* 27, 6 (2017), e1988.

[40] Michele Vadursi, Andrea Ceccarelli, Elias P Duarte, Aniket Mahanti, et al. 2016. System and network security: anomaly detection and monitoring.

[41] Giovanni Venâncio, Rogério C Turchetti, Edson T Camargo, and Elias P Duarte Jr. 2021. VNF-Consensus: A virtual network function for maintaining a consistent distributed software-defined network control plane. *International Journal of Network Management* 31, 3 (2021), e2124.

[42] Giovanni Venâncio, Rogério C Turchetti, and Elias P Duarte. 2019. Nfv-rbcast: Enabling the network to offer reliable and ordered broadcast services. In *2019 9th Latin-American Symposium on Dependable Computing (LADC).* IEEE, 1–10.

[43] Giovanni Venâncio, Rogério C Turchetti, and Elias Procópio Duarte Jr. 2022. NFV-COIN: Unleashing The Power of In-Network Computing with Virtualization Technologies. *Journal of Internet Services and Applications* 13, 1 (2022), 46–53.

[44] Qing Wang, Youyou Lu, Erci Xu, Junru Li, Youmin Chen, and Jiwu Shu. 2021. Concordia: Distributed Shared Memory with In-Network Cache Coherence.. In *FAST.* 277–292.

[45] Yaoguang Wang, Weiming Lu, Bin Yu, and Baogang Wei. 2012. HAaaS: Towards Highly Available Distributed Systems. In *2012 IEEE International Conference on Cluster Computing.* IEEE, 618–621.

[46] Deze Zeng, Nirwan Ansari, Marie-José Montpetit, Eve M Schooler, and Daniele Tarchi. 2021. Guest editorial: In-network computing: Emerging trends for the edge-cloud continuum. *IEEE Network* 35, 5 (2021), 12–13.

[47] Menghao Zhang, Guanyu Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qianqian Li, Mingwei Xu, and Jianping Wu. 2020. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *the 27th Network and Distributed System Security Symposium (NDSS 2020).*