A Survey on Resiliency Techniques in Cloud Computing Infrastructures and Applications

Carlos Colman-Meixner, Chris Develder*, Massimo Tornatore†, and Biswanath Mukherjee
University of California, Davis, USA, *Ghent University – iMinds, Belgium, †Politecnico di Milano, Italy
{cecolmanmeixner, mtornatore, bmukherjee}@ucdavis.edu, *chris.develder@intec.ugent.be, †tornator@elet.polimi.it

***Abstract*—Today's businesses increasingly rely on cloud computing, which brings both great opportunities and challenges. One of the critical challenges is resiliency: disruptions due to failures (either accidental or because of disasters or attacks) may entail significant revenue losses (e.g., US$ 25.5 billion in 2010 for North America). Such failures may originate at any of the major components in a cloud architecture (and propagate to others): (i) the servers hosting the application, (ii) the network interconnecting them (on different scales, inside a data center, up to wide-area connections), or (iii) the application itself. We comprehensively survey a large body of work focusing on resilience of cloud computing, in each (or a combination) of the server, network, and application components.**

First, we present the cloud computing architecture and its key concepts. We highlight both the infrastructure (servers, network) and application components. A key concept is virtualization of infrastructure (i.e., partitioning into logically separate units), and thus we detail the components in both physical and virtual layers. Before moving to the detailed resilience aspects, we provide a qualitative overview of the types of failures that may occur (from the perspective of the layered cloud architecture), and their consequences.

The second major part of the paper introduces and categorizes a large number of techniques for cloud computing infrastructure resiliency. This ranges from designing and operating the facilities, servers, networks, to their integration and virtualization (e.g., also including resilience of the middleware infrastructure).

The third part focuses on resilience in application design and development. We study how applications are designed, installed, and replicated to survive multiple physical failure scenarios as well as disaster failures.

***Index Terms*—Cloud computing, resilience, virtualization, middleware, optical networks, disaster resilience.**

I. Introduction

In the 1960s, the visionary computer scientist John McCarthy predicted the evolution towards service-oriented systems, such as today's cloud computing [1]. Thirty years later, the age of service-oriented computation (SOC) started with the advent of the Internet, that quickly became the worldwide network needed to enable novel cloud services (e.g., email and web services). Convergence of multiple driving forces (i.e., the new SOC paradigm, the rising software industry, high-performance distributed computing, virtualization, and a renewed business interest) paved the road towards the realization of cloud computing [2].

The first widely-available cloud computing service was implemented and commercialized by Amazon in 2006 (EC2/S3 [3]). The concept was standardized by the National Institute of Standards and Technology (NIST) in 2009 [4]. Since then, the importance of cloud computing has risen steadily: Forbes magazine highlighted a market growth of 34% per year and a US$ 19 billion market for 2016 [5]. Yet, with that growth also came the challenges faced by the information technology (IT) industry and service providers to satisfy the increasing demand and diversity of cloud computing services. A major challenge that needs to be overcome to support that growth and thus the massive migration of business services to the cloud is resiliency.

Indeed, even a single physical or software element failure may cause major disruption and severe effects on business revenues [6]: e.g., losses from cloud application and infrastructure failures are estimated at US$ 273 million in 2007–2013, for 28 cloud service providers, given 1,600 hours of disruptions [7].

We define the term *resiliency* as the capacity of a system or an infrastructure or a business to remain *reliable*, *failure tolerant*, *survivable (i.e., recoverable)*, *dependable*, and *secure (i.e., incorruptible)* in case of any malicious or accidental malfunctions or failures that result in a temporal or permanent service disruption [8], [9], [10] (see Glossary in Section VIII).

In this work, we comprehensively survey a large set of studies focusing on resiliency in cloud computing, classified into two major groups: (1) resiliency in infrastructure and (2) resiliency in applications.

Approaches for *resiliency in cloud infrastructure* are designed for the physical and virtualization layers of cloud infrastructure providers. 'Virtualization' means the logical partitioning of physical resources, to increase their utilization by sharing them among multiple users. In a cloud context, 'resources' include processing capacity (located in data centers) and communication network capacity (within the data center, or on a larger, wide-area scale, providing inter-data center connectivity). Earlier surveys on resiliency focused on just a segment of the present-day cloud infrastructure, e.g., specifically for communication networks [11], [12] or data centers [13], [14]. Our survey, on the other hand, provides a holistic view of all cloud computing components and also covers integrated cloud resiliency, focusing on combined communication and computation infrastructures.

The main aim of the proposed approaches for *resiliency in cloud applications* is to reduce the failure's negative impact at the cloud application level, where the service provider interacts with end users. Approaches in this group are classified into: (i) resilient application development, (ii) resilient application collaboration, and (iii) resilient application management.

We will discuss the surveyed resilience studies by considering different aspects. The first aspect concerns the *layers of the cloud system architecture*, with three[1] high-level layers as proposed in [16]: (i) application, (ii) virtualization, and (iii) physical layer. These layers also relate to the service model where applications deliver services over the Internet, and infrastructure, i.e., systems are in data centers [17]. The second aspect is the *type of cloud service disruption*: the cause and severity of the failure that is considered [6], [18]. The third aspect relates to key *additional functionality* provided by cloud computing with respect to traditional SOC (virtualization, multi-tenancy, geo-distribution and ubiquitous access, dynamic resource provisioning, self-adaptability, etc.) [2], [15]. Finally, we consider the fundamental *resiliency techniques* to counter failures (e.g., forecasting, pre-provisioning of extra capacity for protection, and post-failure recovery).

Section II starts by clarifying the *cloud architecture* and associated terminology. Section III gives a general overview of *cloud service disruptions*, and introduces the terminology used to describe their causes, severity, and financial consequences. In Section IV, we explain the main conceptual solutions to those disruptions, to come to a *classification of resiliency strategies* that we subsequently adopt to survey the studies detailing the solutions. We partition the studies in two top-level categories: Section V discusses the *infrastructure resilience*, while Section VI covers *application-layer resiliency*. Section VII concludes the paper with a summary of the major approaches (and how they compare), an analysis of major trends, and open problems in cloud resiliency followed by a glossary of important terms in Section VIII and references.

II. Cloud Computing Concepts and Definitions

Here, we first summarize the cloud service model (Section II-A) and functionality that cloud computing provides beyond traditional service oriented computing (Section II-B). Readers familiar with cloud computing can directly move to our model and terminology of the architectural layers in Section II-C.

A. Service model

The cloud computing service model is usually referred to by an acronym "XaaS" (X-as-a-service, where X can be any offered cloud service) [16], [17]. Among the numerous Xs, three baseline services are defined by NIST: (1) *Software-as-a-Service (SaaS)*, that provides access to online software applications to one or more tenants and/or end users (e.g., Dropbox, Google Docs); (2) *Platform-as-a-Service (PaaS)*, where a platform such as an operating system or an application development framework is provided to users (e.g., Amazon Web Service (AWS), Salesforce.com, MS Windows Azure); (3) *Infrastructure-as-a-Service (IaaS)*, where, of a given cloud network infrastructure, a fraction of the connectivity, processing, and storage capacities are provided to one or many tenants (e.g., the EC2 cloud of Amazon [3], [16], [19]). According to [15], a PaaS provider's cloud is very likely to run on top of an IaaS provider's cloud, and many PaaS and IaaS providers tend to be the same organization. For these reasons, in this survey, we refer to a SaaS provider as *"cloud application service provider"* and to PaaS and IaaS provider as *"cloud infrastructure service provider"*, as suggested in [17].

[1]Note that some previous surveys have also suggested a four-layer model of application, platform, infrastructure, and hardware [15]. However, we choose the three-layer model suggested in [16] for the sake of simplicity.

B. Cloud functionalities beyond traditional SOC

Functionalities that cloud computing provide with respect to traditional service-oriented computing include the following [4]:

1. *Virtualization* provides physical resource partitioning with adaptability and privacy. Virtualization is largely adopted to achieve resource sharing, with isolation, and enables Migration (see Glossary in Section VIII). There are three levels of virtualization: (i) application, (ii) operating system, and (iii) hardware [20]. Virtualized applications imply separate user spaces with specific isolated application instances (i.e., sessions). Operating-system level virtualization creates independent and isolated operating systems with all functionalities and attributes (i.e., virtual machines, VM). Hardware virtualization is the slicing of hardware resources (e.g., virtual optical networks) [20] (see Glossary in Section VIII).

2. *Multi-tenancy* is the possibility for multiple service providers to share the physical infrastructure of a single operator and/or for one service provider to use physical infrastructure of multiple operators [15]. Multi-tenancy can be adopted to increase resiliency (e.g., SecondSite [21]), assuming that failures affecting multiple providers are unlikely.

3. *Dynamic resource provisioning* allows resources to be leased and released from a pool, whenever needed. Most approaches for resiliency in physical infrastructure exploit it (e.g., Self-Adaptive Cloud Collaboration (SACC) [22]).

4. *Geo-distribution and ubiquitous access* allow services and data to be accessed from anywhere. These characteristics are shared with SOC and are also largely employed in support of disaster resiliency, e.g., when moving content to avoid massive disruption due to an expected disaster (e.g., DeepSky [23]).

5. *Self-adaptability* (i.e., self-migration and self-reorganization) consists in the leasing and releasing of resources on demand from cloud providers by using automated resource-management features to reorganize their cloud computing infrastructures and respond efficiently to rapid changes in service demand (e.g., flash-crowd effects). Some approaches exploit this functionality by offering automatic failure mitigation in cloud computing operations (e.g., NetPilot [24]).

C. Layered cloud architecture

We now introduce the three-layer cloud architecture suggested in [16] and the taxonomy of components presented in [25]. Figure 1 illustrates the following layers, with sample organizations whose core businesses relate to the respective layer:

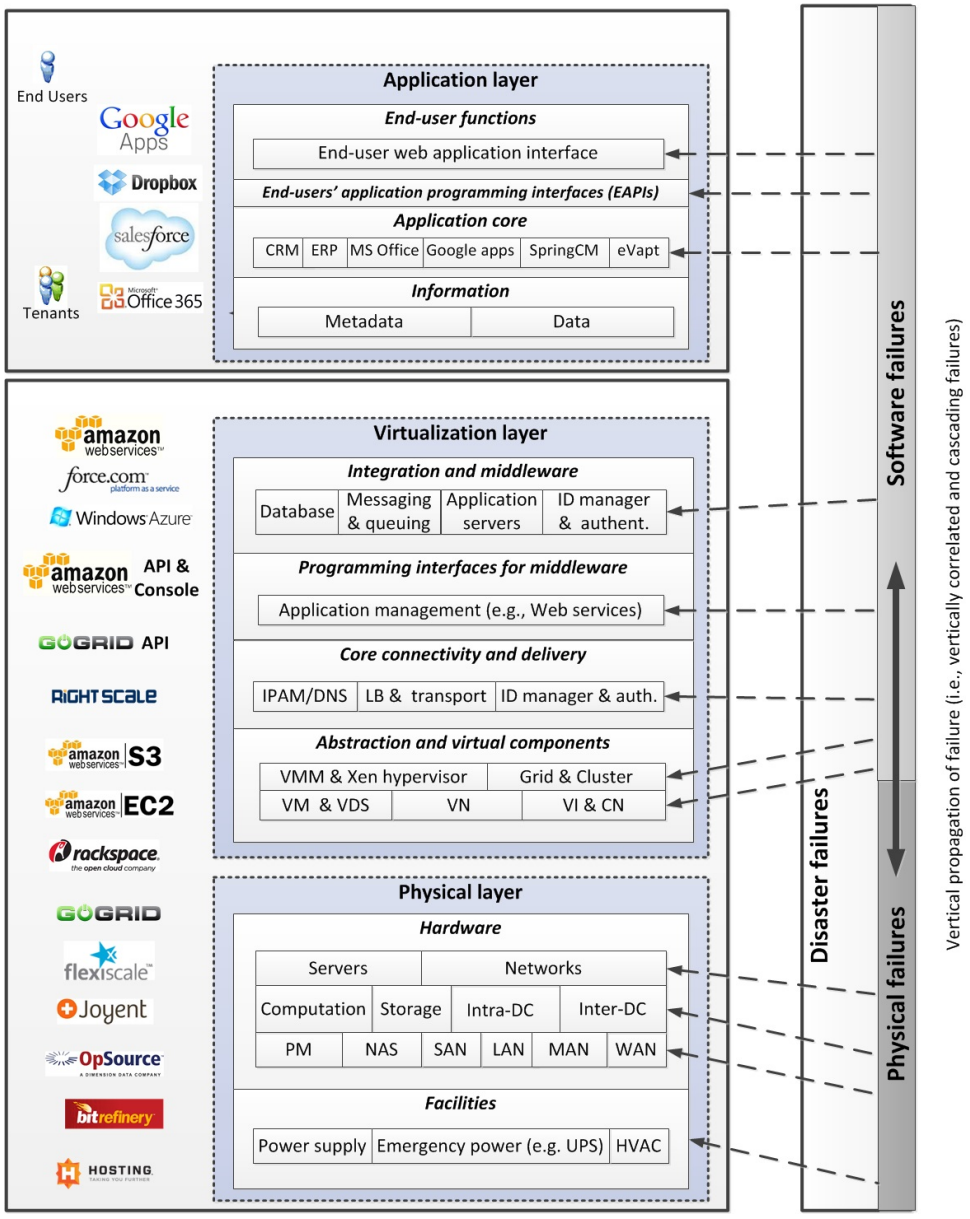

Fig. 1. Cloud computing and cloud service disruption.

1. The *application layer* is the upper-most layer of the hierarchy where data and applications reside, and has four building blocks [25]:

1.a) *End-user functions* include end users' and tenants' interfaces and functionalities, e.g., web interfaces;

1.b) *End-users' application programming interface (EAPI)* defines an interface for development and customization of users' or tenants' applications;

1.c) *Application core* includes the engine of each cloud application, e.g., Customer Relation Management (CRM), Enterprise Resource Planning (ERP); and

1.d) *Information* manages two different granularities of information, *data* and *metadata* [25].

2. The *virtualization layer* supports the application demands by organizing, managing, and providing access to resources of the physical layer, and comprises four building blocks [25]:

2.a) *Integration and middleware* components provide the infrastructure with application development frameworks, middleware capabilities, and useful functions such as database access, message interchanging, and queuing (e.g., a MapReduce interface). This building block also provides basic server functionalities for fundamental applications (e.g., email, web) and secure authentication functionalities (e.g., identity manager and authentication protocols) as a platform.

2.b) *Programming interfaces for middleware* comprise a set of APIs for IaaS tenants to support their interaction with

the infrastructure (e.g., the Restful Web Service API used by Amazon and GoGrid).

2.c) *Core connectivity and delivery* manage the connectivity to the infrastructure and define virtual topologies: Internet Protocol (IP) address management (IPAM) for internal and external IP addresses; Domain name system (DNS) for translation of IaaS tenants' IPs and domains (e.g., automatic AWS DNS name assignment); load balancers (LB) and transport which distribute workloads across servers, network links (e.g., AWS Auto-Scaling); identity access management (e.g., AWS credentials and X.509 certificates) functionality with network firewalls to control the access and provide security for the infrastructure and IaaS tenants' data.

2.d) *Abstraction and virtual components* provide abstraction and virtualization of physical infrastructure for the tenants' platforms and applications. This building block defines and controls the virtual machines (VMs) and virtual digital sites (VDS) by using hypervisors (virtual machine manager, VMM, and Xen), and maps virtual networks (VNets/VNs) integrated in cloud networks (CNs) over a substrate physical infrastructure (see Glossary in Section VIII).

3. The *physical layer* provides physical resources for computation, communication, and storage data by defining two building blocks:

3.a) *Hardware* comprises the core data center equipment, i.e., a pool of servers or physical machines (i.e., PM) connected using an intra-data center network composed of, e.g., switches and routers. Cloud infrastructure providers typically manage a network of data centers, connected by core networks (i.e., the inter-data center network).

3.b) *Facilities* refer to the supporting equipment comprising power systems and HVAC (heating, ventilation, and air conditioning).

Figure 2 shows three cloud infrastructure providers managing their virtualized infrastructure (i.e., servers, networks, data centers, networks of data centers, and facilities) by providing geographically-distributed and ubiquitous access to VMs for multi-tenant cloud applications. Some tenant's applications are for private usage, while others provide services for end users (i.e., SaaS).

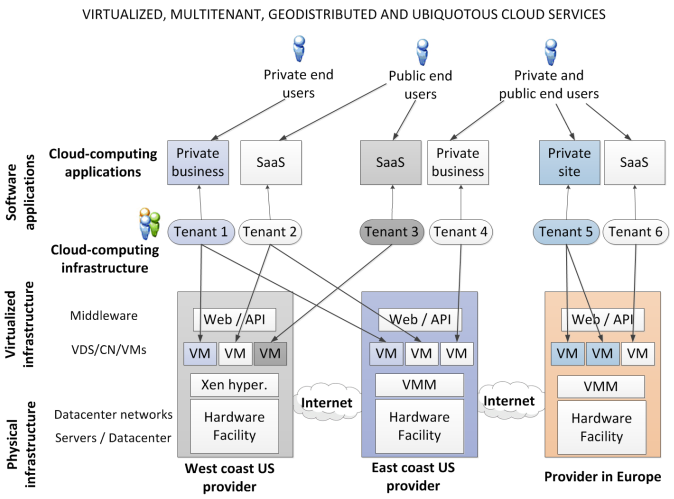

Fig. 2. Three cloud infrastructure providers in different geographical zones (i.e., US west coast, US east coast, and Europe); six tenants (i.e., SaaS application services and private business and sites); and two types of end users, using virtualization, multi-tenancy, and geo-distributed and ubiquitous access functionality of cloud computing.

III. Cloud Service Disruption

Before we detail the actual studies in subsequent sections, we first provide an overview of the causes of cloud service disruptions, the severity of the impacts, and their associated costs.

A. Causes of cloud service disruptions

Figure 1 illustrates what layers the various main causes have an effect on. The main causes can be characterized as follows:

1. *Human errors:* failures induced by human actions, unintentionally or intentionally. When produced unintentionally, the error can be typically controlled and fixed (e.g., a cloud operator accidentally erases the profile of a user or shuts down some vital components). However, some errors can be intentionally produced by attacks or sabotage to destroy or disrupt cloud services and data [26].

2. *Software failures:* caused by software bugs, software aging, errors in the application design, and malicious software (e.g., a virus). Grottke *et al.* [27] present a taxonomy of these failures.

3. *Physical failures:* physical component outages. These failures propagate vertically along the cloud system hierarchy (i.e., they start from the physical layer and propagate to the virtualization and application layers). Such failures can also propagate horizontally, affecting multiple hardware components (e.g., a failure of a server may produce post-failure traffic overload, subsequently failing more servers).

4. *Disasters:* natural disasters (tornadoes, earthquakes, etc.) or human-induced ones (e.g., weapons of mass destruction, WMD). These failure scenarios often have a high impact, as they can propagate to the higher layers with a cascade of physical failures, software failures, and human errors.

B. Impact of cloud service disruptions

We classify the service disruption severity in four levels of increasing impact [18]:

1. *Masked disruption*, when the service provider can recover failed components without causing service disruption for end users and tenants.

2. *Degraded service*, when offered services get reduced in quality, but are not interrupted. Some degradations may incur penalties (see next subsection).

3. *Unreachable service*, when the service or data still resides in the servers, but cannot be accessed by end users or tenants. This is the most common case for network communication failures.

4. *Corrupted service*, when the service cannot be recovered and end users and tenants lose critical data and associated

revenues and might initiate legal actions against the cloud service provider.

C. Consequences of cloud service disruptions

Cloud service providers, tenants, and users confront three different costs or losses induced by cloud service disruption:

1. *Repair costs:* paid by the cloud service provider when failures affect cloud components. Vishwanath *et al.* [28] provide a quantitative evaluation of the effect of failures in servers of large data centers. The study observed failures and replacement of servers of about 8% of 100,000 servers (i.e., 8,000 servers) in a period of 14 months, amounting to a total repair cost (without considering penalty) of around US$ 2.5 million. Similar results are reported in another study performed on the Google cloud infrastructure [18].

2. *Penalty costs:* incurred by cloud application and cloud infrastructure providers when cloud services are disrupted by reducing their availability. As an example, the availability and penalty costs incurred by 28 cloud service providers due to service disruptions in 2007–2013 were estimated to amount to over US$ 273 million [7].

3. *Business revenue losses:* the missed business opportunities, in an extended sense, due to cloud service outages. This can be very significant, given the dependency of today's business on cloud computing services: [29] estimates a loss of US$ 25.5 billion for the year 2010, considering total or partial service disruptions in North American businesses.

IV. Classification of Resiliency approaches in Cloud Computing

We now present the classification of resiliency in cloud computing, as adopted in the rest of the paper.

A. Resiliency strategies

Starting from the traditional classification of network resilience [11] and information technology resilience [9], we develop a comprehensive classification to cover all resiliency methods in cloud systems (i.e., methods applicable to cloud infrastructures as well as cloud applications). Our comprehensive classification comprises failure forecasting and removal, protection, and restoration:

1. *Failure forecasting and removal:* consists in using measurements and/or estimations of possible failures and their consequences for the system to enhance system preparedness. For cloud infrastructure providers, failure forecasting and removal imply the reorganization of their infrastructure or software components to reduce or remove negative effects of failures before they occur. Degrading service can be considered as a removal approach because no additional capacity is pre-provisioned.
An example of failure forecasting and removal is resilient virtual machine migration using traffic demand estimation.

2. *Protection:* consists in pre-deploying redundant computational and communication capacity to recover the services in case a failure occurs. An example of these strategies applied for cloud computing resiliency is the PipeCloud approach proposed in [30], which replicates entire tenant sites to provide protection.

Protection in distributed systems can be implemented through replication- and checkpoint-based methodologies [31]:

2.a) *Replication* is the most common protection method and consists in the total or partial duplication of capacity (e.g., data, connections, machines, etc.) to provide protection in case of failure. The replication can be active or passive. *Active* replication keeps many replicas updated (e.g., N-version programming [32], RAID for disk storage, and $1{:}N$ protection for communication networks [11]). The *passive* scheme defines a working component and a dedicated backup replica where the backup is activated in the case of failure of the working component to avoid disruption (i.e., recovery block [33], dedicated-path protection or 1:1 protection, and shared-path protection [11]).

2.b) *Checkpointing* consists in periodically saving the system state in a different place, and restoring it from there if the original instance fails. An example of this method is the dynamic adaptive fault-tolerant strategy [31].

3. *Restoration (a.k.a. recovery)* refers to reactive and best-effort schemes to reduce the impact of unexpected failures after they occur (as opposed to protection where redundant capacity is pre-deployed). A common recovery strategy for objects, applications, and components is the retry and reboot method, which consists in reloading the failed application or component [34]. A well-known strategy in networks is best-effort traffic re-routing and capacity re-provisioning to recover from physical network failures.

B. Metrics for resiliency objectives

The cloud computing resiliency approaches in this survey use very diverse metrics and objective functions to quantify and optimize their efficiency to reduce the severity or duration of cloud service disruption. To provide an overview, we classify metrics and objective functions based in three main areas:

1. *Survivability* is the capability of a system to survive a certain kind of failures (i.e., single, multiple, and disasters) and attacks (i.e., WMD and Byzantine) [8] (e.g., recovery point objective (RPO) [35]).

2. *Recovery time (RT)* is the duration of the failure recovery phase, typically used synonymously with mean time to repair (MTTR). The maximum tolerated RT is the recovery time objective (RTO), which is listed as part of the service-level objective (SLO) [35].

3. *Cost* in resources associated with the resiliency scheme include CAPEX, OPEX, penalty, and revenue losses.

C. Categorization of surveyed cloud resiliency approaches

Following the service model from Section II-A, we categorize the surveyed approaches in two main groups, discussed in detail in Sections V and VI, respectively:

1. *Resiliency in cloud computing infrastructures* contains a group of approaches designed for cloud infrastructure

providers (i.e., IaaS and PaaS) that interact with the virtualization and/or physical layer, categorized in five types of infrastructure: facility, servers, networking, cloud integrated infrastructure, and cloud middleware.

2. *Resiliency in cloud computing applications* contains a group of approaches designed for cloud application containing providers (i.e., SaaS), especially in interaction with cloud applications and contents.

V. Resiliency Techniques for Cloud Computing Infrastructure

A cloud infrastructure provider designs, implements, and operates one or more data centers (i.e., data center facilities) interconnected by (inter- or intra-data center) communication facilities. We categorize the resiliency techniques for the cloud infrastructure in six groups (Fig. 3(c)).

The first two categories (A, B) deal with resiliency inside the data center (computing) infrastructure, and specifically with *resiliency of facilities* (e.g., power, HVAC, etc.; see Section V-A) and *resiliency of servers* (see Section V-B). The next two categories (C, D) cover the resiliency in inter-data center communication networks (*resiliency of networks*, Section V-C) and resiliency of the integration of the communication network with the servers (*resiliency in cloud integrated infrastructures*, Section V-D). The final groups (E, F) focus on higher-level components (*resiliency of middleware*, Section V-E) and some measurement and estimation techniques for resiliency (*resilient measurement and estimation*, Section V-F).

A. Resilient facility design and operation

The primary concern for a cloud computing infrastructure provider is the resiliency of the data center facility to physical threats (i.e., disaster, sabotage, terrorist attacks, human errors, hardware trip, energy blackouts, WMD attacks) and logical threats (i.e., hackers and cyber terrorism). Figure 4 presents the elements of a data center facility. Resiliency targets for data center facilities are classified in tiers, ranging from tier 1 (looser requirements) to tier 4 (stricter requirements) [36]. To reach a higher tier, the cloud provider must provide resiliency at the level of the power systems, facility access, and facility operation, as discussed below.

1. *Power resiliency:* is the foundation of any cloud facility given its critical dependency on energy supply.

1.a) *Power redundancy:* (or "emergency" power) protects the facility against temporary power outage given the vulnerability of the power grid to accidents, weather conditions, and malicious attacks [37], [38]. Two elements are recommended to minimize the impact of power outages: *uninterruptible power supply (UPS)* and *emergency power generation* (e.g., diesel engine) [18].

1.b) *Resiliency against power quality issues:* to protect the facility due to sudden power fluctuations, spikes, or surges, two elements are highly recommended: the *secured power distribution unit (PDU)* and *high-voltage DC distribution (HVDC)* [39], [40].

1.c) *Resiliency against heat accumulation:* redundant and protected cooling systems to avoid devices from overheating (e.g., cooling systems) [18], [41].

2. *Resilient facility access:* techniques that protect cloud infrastructure facilities from physical and cyber access of malicious users [25], [42].

3. *Resilient facility operation:* the implementation of training and business continuity plans for prevention, detection, mitigation, and recovery in case of disasters or attacks [25], [42].

Note that most resiliency techniques for facility design and operation are designed and tested by a multidisciplinary set of hardware and facility industries with the support and standardization of diverse organizations [9], [10], [25], [36], [42].

However, resiliency techniques for cloud infrastructure introduced in the following sections (V-B through V-F) were mostly proposed and tested by academia, typically with support from the information technology (IT) industry (i.e., hardware and software industries).

B. Resiliency in servers

Physical servers are essential for computation and storage in cloud computing infrastructures. Resiliency techniques in servers are implemented at both the physical and/or the virtualization layer (Fig. 5).

The surveyed approaches in server resiliency can be divided in two groups: *Resiliency in physical servers* covering resiliency techniques for physical servers and storage units; *Resiliency in virtual servers* covering resiliency techniques for virtual machines (VMs) and virtual storage (VS) (see Glossary in Section VIII). Figure 5 introduces the taxonomy for resiliency-in-servers techniques, while Table I summarizes and compares the approaches discussed in this subsection.

1. *Resiliency in physical servers:* covers a set of failure removal, protection, and recovery methods acting at different levels, namely the physical machine (PM) components, the processes, and the data inside the physical server.

In this regard, the *physical-failure isolation or fencing* method provides failure removal in PM components and processes, while the *process-level replication (PLR)* method enforces failure protection and *process checkpointing* provides failure recovery in process level [43], [44], [45]. Protection and recovery of data are enforced by *error detection and correction coding (EDCC)* and *redundant array of independent disks (RAID)* techniques.

1.a) *Failure isolation or fencing:* is the isolation of failed circuits or components in the hardware or process in the software to avoid failure propagation. An example of fencing used in multiprocessor architecture is the *failure contention and component retirement* in [46].

1.b) *Process-level replication (PLR):* consists in running multiple executions of the process on different cores or CPUs (OS level) [44], and/or in different servers and/or in different cloud providers [45].

1.c) *Process checkpointing:* is the re-initialization of a process, or of a server, or of the entire OS when failures

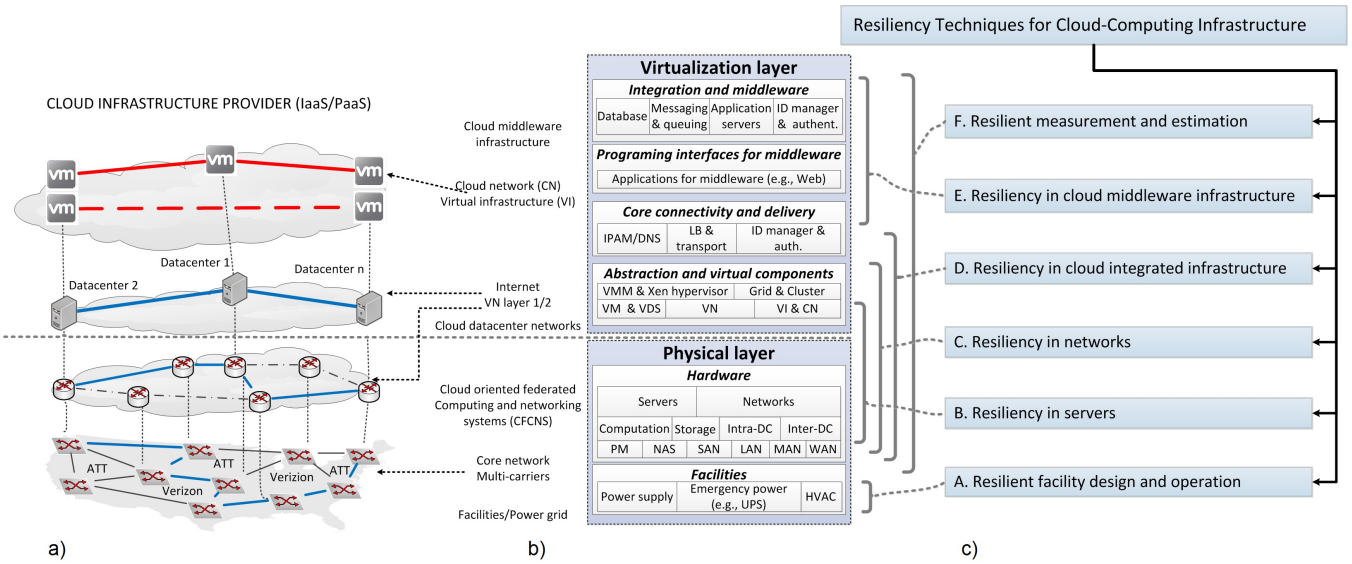

Fig. 3. A high-level view of the cloud computing infrastructure provider and taxonomy of resiliency techniques in cloud infrastructure: a) Example of a cloud computing infrastructure where integration between computing and communication resources is provided; b) The two layers of a cloud computing infrastructure with building blocks and their association with infrastructure layers in a); c) Taxonomy of resiliency techniques for cloud infrastructures and their association with layers in b).

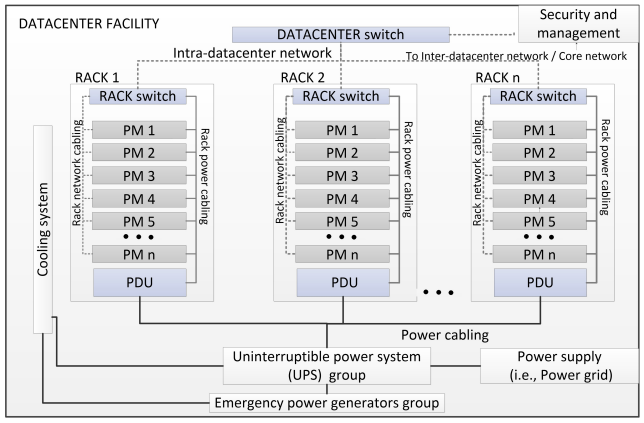

Fig. 4. Cloud data center facility, racks, intra-data center network, power system, and cooling system.

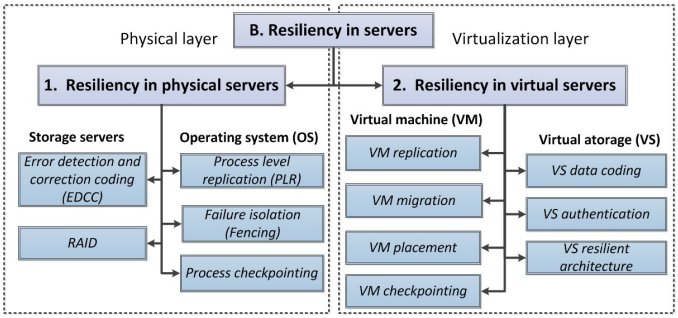

Fig. 5. Taxonomy of resiliency techniques for servers.

or abnormal functions occur. Management tools for cloud infrastructures use reset/reboot to deal with failed process in PMs and VMs [47].

1.d) *Error detection and correction coding (EDCC):* is a technique used in memories and large storage devices. For memory (i.e., DRAM), *one bit correction and two bits detection* [48] uses a periodic parity check to detect and correct one bit per byte. For larger storage devices, there are two common coding techniques: the hamming code (HC) [49] and the Reed-Solomon (RS) code [50]. A major challenge for EDCC is the computing capacity required to process a large volume of data.

1.e) *Redundant array of independent disks (RAID):* combines EDCC technique with data redundancy to enforce low-cost active and passive disk replication (i.e., mirroring). RAID stripes the data in bytes or blocks with single or double parity EDCC (i.e., single or multiple dedicated parity disk). The cloud infrastructure provider may install and configure its storage devices using one or a combination of six types of RAIDs. A summary and comparison of the various RAID types can be found in [51].

2. *Resiliency in virtual server:* includes resiliency methods based on server virtualization acting on virtual machines (VM) or on virtualized storage (VS). Virtual server protection is provided by *VM replication* in different servers, while VS recovery is provided by *VM migration* and *VS data coding*. By combining VS protection, recovery and failure removal, some approaches use *VM checkpointing* and *VM placement* techniques. VS resiliency techniques focus on data recovery and/or protection, but also include data integrity enforcement by *VS authentication* for data integrity control, and *VS resilient architecture design* to unify data coding and authentication

Table I
APPROACHES FOR RESILIENCY IN SERVERS

Techniques and approaches				Contribution for cloud infrastructure resiliency / Type and level of resiliency enforced for servers							
Type	Approach/es	Resiliency technique	Method	Benefits	Challenges	PM	VM	Software*	Physical†	Attack€	Disaster††
	Physical layer										
P/R	PLR [43], [44], [45]	Process replication	PR	Fast recovery	Additional resources	Single	-	+	+	-	-
R/R	Fencing [46]	Failure isolation	RM	Avoid failure propagation	No recovery functions	Partial	-	+	+	-	-
RUN	Reset/reboot [47]	Process checkpoint	RE	Low cost and RT	Processing overhead	Single	-	+	+	+	-
RUN	EDCC [48]	Bit correction	RE	Data recovery	Processing overhead	Partial	-	+	+	+	-
P/R	RAID	Data replication	RE/PR	Data recovery	Processing overhead	Single	-	++	++	+	-
	Virtualization layer										
P/R	Zap [52]	VM replication	PR	Fast VM recovery	OS diversity and cost	Single	Single	+	+	-	-
P/R	AutopoD [53]	VM replication	PR	Fast VM recovery	Large VMs and cost	Single	Single	++	+	-	-
P/R	FT-VM [54]	VM replication	PR	Fast multi-VM recovery	Large VMs and cost	Single	Multiple	++	+	-	-
RUN	Xen-VM-Mig [55]	VM migration	RE	Low cost VM recovery	Large VM RT	Single	Single	++	+	-	-
RUN	Vmotion [56]	VM migration	RE	Low cost VM recovery	Large VM RT	Single	Single	++	+	-	-
RUN	PostCLVM [57]	VM migration	RE	Low VM RT and downtime	Large service diversity	Single	Single	++	+	-	-
RUN	P-FT [58]	VM placement	PR	Low VM RT and downtime	Traffic losses	Single	Single	++	+	-	-
RUN	RDVM [59]	VM placement	PR	Low VM RT and downtime	Traffic losses	Single	Single	++	+	-	-
RUN	SCVP [60]	VM placement	FR/RE	Low VM RT and traffic losses	Large service diversity	Single	Single	++	+	+	-
RUN	AFRC [61]	VM placement	PR	Low VM RT and traffic loses	Large service diversity	Single	Single	++	+	+	+
P/R	HA-VM-Place [62]	VM placement	RE/PR	Multi-VMs with low RT	Large service diversity	Multiple	Multiple	+++	++	+	-
P/R	HA-SBS-Place [63]	VM placement	RE/PR	Multi-VMs with low RT	Large service diversity	Multiple	Multiple	+++	++	+	+
P/R	HA-LOC [64]	VM placement	PR	Multi-VMs with low RT	Large service diversity	Multiple	Multiple	+++	++	+	-
RUN	ISR [65]	VM checkpointing	RE	Low VM RT and cost	Storage space usage	Multiple	Single	+++	++	+	-
RUN	Remus [66]	VM checkpointing	RE	Low VM RT and cost	Checkpoint data update	Multiple	Single	++	++	+	-
RUN	KEMARI [67]	VM checkpointing	RE	Low VM RT and cost	Checkpoint data update	Multiple	Single	++	++	+	-
RUN	FVMCheck [68]	VM checkpointing	RE	Low VM RT and cost	Storage space usage	Multiple	Single	+++	++	+	-
RUN	RemusDB [69]	VM checkpointing	RE	Low VM RT and cost	Large service diversity	Multiple	Multiple	+++	++	+	+
RUN	RS [50]	VS coding	RE	Low data RT and cost	Large cost	Single	Single	+	+	-	-
RUN	SRS [70]	VS coding	RE	Low multi-data RT and cost	Large service diversity	Single	Single	++	+	-	-
RUN	ByzantCode [71]	VS coding	RE/PR	Byzantine attacks mitigation	Processing overhead	Single	Single	++	++	++	-
RUN	PoWerStore [72]	VS authentication	RE/PR	Data consistency	Byzantine attacks	Single	Single	++	+	-	-
PLA	HAIL [73]	VS design	RE/PR	Data recovery	Large cloud storage	Multiple	Multiple	++	++	+	-
PLA	Depot [74]	VS design	RE/PR	Data recovery	Large clouds storage	Multiple	Multiple	++	++	++	-

Techniques and approaches
Type: type of utilization by the cloud provider. "PLA" infrastructure planning (e.g., placement and dimensioning). "RUN" run-time. "P/R" the approach can be used in both phases.
Method: "PR" protection, "RE" recovery, "FR" forecasting, "RM" removal, "RE/PR" either recovery or protection or both.
Level of resiliency
*Survivability to different extent of software failures, "+" failure of some processes and data, "++" failure of multiple processes and data.
†Survivability to different extent of hardware failures, "+" failure of some components of the PM and storage, "++" failure of multiple components and storage devices.
€Survivability to different extent of attacks, "+" attacks with easy recovery, "++" attacks affecting some components with data losses, "+++" attacks affecting many software and hardware component with important data losses.
††Survivability to different level of disasters, "+" disaster without cascading failures affecting a set of PM.

techniques (Table I).

2.a) *VM replication:* consists in replicating VMs in multiple PMs to provide resiliency. VM replication must ensure replication diversity (of PMs and OS environment), synchronization, and low cost in terms of additional resources. Earlier VM replication techniques, such as *Zap* [52], only work among homogeneous OSs and similar PMs. Later, the approach *AutoPod* [53] was suggested to achieve VM replication between heterogeneous OSs and PMs. To address the challenge of replica synchronization, a recent extension of Zap and AutPod, called *fault-tolerant VM (FT-VM)* [54] has been proposed that is able to save up to 10% of the synchronization overhead compared to baseline replication schemes. VM checkpointing and VM placement approaches (discussed later) have been suggested to minimize cost while providing the fast recovery of VM replication.

2.b) *VM migration:* consists in the relocation of the entire VM content from a failed PM to a non-failed PM. This scheme provides low-cost, yet best-effort recovery [67], [75]. Two approaches to VM migration can be used: pre-copy and post-copy. Pre-copy migration relocates the full VM content (i.e., memory and storage) before initiating the users' reconnection, threfore inducing significant service downtime. Two such approaches are *Xen VM migration (Xen-VM-Mig)* [55] and *VMotion* [56]. VMotion achieves lower service downtime compared Xen-VM-Mig, as it does not interrupt the access to the VM during the migration process. Post-copy migration techniques gradually replicate the content of the VM without interrupting the communication from users. One approach using this technique is *PostCLMV* [57] which reduces the migration time by 50% compared to Xen-VM-Mig and VMotion. However, the service downtime is still high for large and diverse VMs [67] compared to VM replication, VM placement, and VM checkpointing (see Table I).

2.c) *VM checkpointing:* periodically copies the last working state of the VM in different PMs to provide fast recovery. An early VM checkpointing approach is the *internet suspend/resume (ISR)* for VMWare [65]. Two more recent extensions of ISR for VM replication are *Remus* [66] and *KEMARI* [67]. The main drawbacks of VM checkpointing are the synchronization overhead and the storage space required for a larger number of VMs. In this regard, *similarity compression (SC)* [76] reduces the storage space by using a compression algorithm, while *fast and capacity-efficient VM checkpoint (FVMCheck)* [68] reduces the storage space and synchronization overhead using page-caching technique. More recently, an extension of Remus, *RemusDB* [69], has been proposed to reduce synchronization and checkpoint overhead (see Table I).

2.d) *VM placement* includes those techniques that try to obtain a resilient allocation of VMs in the cloud infrastructure by intelligent placement [77]. We surveyed three resilient VM placement techniques: pre-migration, clustering of VM replicas, and resilient VM allocation. The pre-migration technique consists in predefining a migration location for a VM in case of PM failure, by reserving an empty VM ready to receive an emergency evacuation of the working VM. This technique reduces the migration time of baseline VM migration techniques, however the determination of the point for migration is challenging. The *proactive-fault tolerance (P-FT)* approach in [58] uses a failure-prediction technique based on the failure history of each server. Significant reduction of downtime during migration is achieved by *re-configurable distributed virtual machines (RDVM)* [59], [78]. Multiple server failures are addressed by *high available VM placement (HA-VM-Place)* [62] and *VM shadow-based placement solution (VM-SBS-Place)* [63]. Another important aspect in the resilient VM placement problem relates to clustering. In fact, clustering of VM replicas allows to distribute VMs in different PMs to avoid single point of failures. Approaches using this technique are *structural constraint aware VM placement (SCVP)* [60] for single-server failures, and *high available logical encapsulation (HA-LOC)* [64] for multiple server failures. Finally, resilient VM placement can be also achieved based on the specific (and diverse) resiliency demands of each application, as in *adaptive fault tolerance in real-time cloud computing (AFTRC)* [61], developed to minimize downtime (i.e., recovery time (RT)) of real-time applications.

2.e) *VS data coding:* consists in the use of data coding techniques for recovering data of virtualized storage (VS) in case of physical failures or Byzantine attacks. For data recovery in case of server failures, a *scalable Reed-Solomon (SRS)* approach is suggested in [70] to reduce the processing time, storage space, and bandwidth required by the canonical Reed-Solomon (RS) technique [50]. However, as Byzantine attacks might corrupt the data coded by SRS, a *Byzantine coding (Byzant-Code)* technique is suggested in [71] to defend data from such attacks (see Glossary in Section VIII).

2.f) *VS data authentication:* ensures data consistency to prevent data losses and anomalies during failures or Byzantine attacks into VSs (e.g., *proofs of writing (PoWerStore)* protocol suggested in [72]).

2.g) *VS resilient architecture designs:* suggest the addition of a specialized layer into the VS or cloud storage system to enforce resiliency. Two proposals for VS design are *high-availability and integrity layer (HAIL)* [73] and *cloud storage with minimal trust (Depot)* [74]. HAIL adds a layer with a proof-of-retrievability (POR) module in each server to mitigate data losses due to server failures or Byzantine attacks (see Glossary in Section VIII). Depot proposes a two-layer VS architecture, where the first layer enforces data replication, while the second layer implements protocols for data consistency and recovery functionalities (e.g., PowerStore, Byzant-Code). Table I confirms the ability of the approaches in this category to combine the benefits of coding and authentication techniques.

Discussion of benefits and challenges for resiliency techniques in servers: Table I presents the key benefits and challenges of resiliency in servers with a brief comparison of the levels of survivability achieved. The most important benefits are the high redundancy, low cost (e.g., VM mi-

gration approaches), and low recovery time (RT) (e.g., VM checkpointing). The main challenges derive from large service diversity in cloud computing, large run-time complexity, and large volume of data or content placed in the infrastructure. As a result, some resiliency techniques face problems of high computational complexity (e.g., multiple VM placement problem, multiple VM checkpointing synchronization, and high data coding overhead in VS servers) that require efficient optimization tools or further studies. Indeed, in terms of resilient planning and management, techniques for resiliency in servers are designed to support resiliency planning (i.e., PLA) and to be integrated with management tools for run-time or operational resiliency (i.e., P/R, RUN). However, they depend on the network availability and an efficient resource allocation to deal with cloud infrastructure complexity. Hence, techniques for resiliency in servers are integrated with others techniques to deal with cloud infrastructure resiliency (Sections V-C, V-D, V-E and, V-F). Before we advance, we will discuss techniques for resiliency in networks which are essential for the integration of techniques for resiliency in cloud infrastructure.

C. Resiliency in networks

The resiliency of the communication network is essential for cloud infrastructures that are typically geo-distributed. The network infrastructures used by cloud providers are two: the intra-data center network (Intra-DCN) using a local-area network (LAN) technology inside a data center facility, and the inter-data center network (Inter-DCN) using wide-area network (WAN) technologies (e.g., optical networks) interconnecting two or more data centers into a federated cloud. Besides the design and operation of their own data center facilities, cloud providers might use communication services from carriers and/or Internet services providers (e.g., AT&T) and/or manage their own large network communication (i.e., WAN) infrastructure (e.g., Google). On top of such large communication networks, network virtualization is enabled to enforce multi-tenancy, geo-distribution, and resource pooling as discussed in Section II-B.

Two surveys [11], [115] have presented and classified important studies on communication network resiliency. A survey about disasters and their effects in optical networks is presented in [12]. However, virtual network environments tends to be particularly prone to failure disruptions as the failure in one single physical element (i.e., an optical link) might disconnect a large set of virtual connections from different tenants and cloud applications. Some techniques for virtual-network resiliency (e.g., survivability of virtual network embedding (SVNE)) were surveyed in [116]. Here, we extend and complement the existing overviews with a specific focus on the resiliency techniques for data center networks and virtualized networks that are relevant from the perspective of a cloud provider. Figure 6 introduces the taxonomy of resiliency techniques, and Table II presents our categorization for network resiliency in cloud infrastructure.

1. *Resiliency in data center networks:* includes techniques for resilient design and traffic engineering for intra-data center networks (Intra-DCN) and their extension for the inter-data center network (Inter-DCN). When techniques are applicable to both inter- and intra-DCN scenario, we simply use the term DCN.

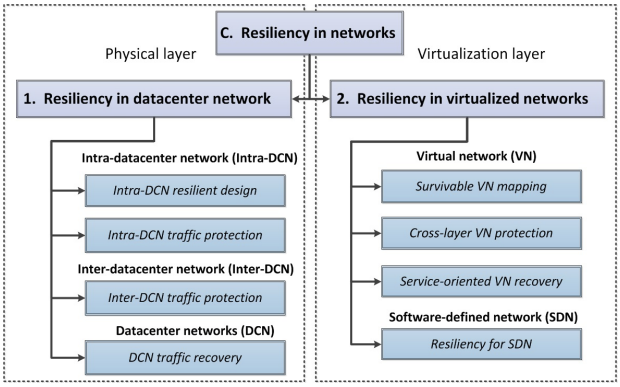

Fig. 6. Network resiliency taxonomy.

1.a) *Intra-DCN reliable design:* should complete four important requisites to increase the network resiliency: high nodal degree, existence of backup routes, large critical cuts (see Glossary in Section VIII), and high capacity for redundancy (i.e., for protection and restoration). Three design strategies were suggested for meeting these requirements:

Switch redundancy consists in the addition of backup top-of-rack (ToR) and/or aggregation (Agg) and/or core switches in Intra-DCNs [117]. Some well-known architectures enabling this scheme are: *FatTree* [79], *VL2* [80], and *Qfabric* [81]. The Qfabric architecture provides the highest survivability, but with highest cost, while less-costly but less-resilient Intra-DCN architecture are shown in Table II. Figure 7(a) shows an example of the FatTree scheme.

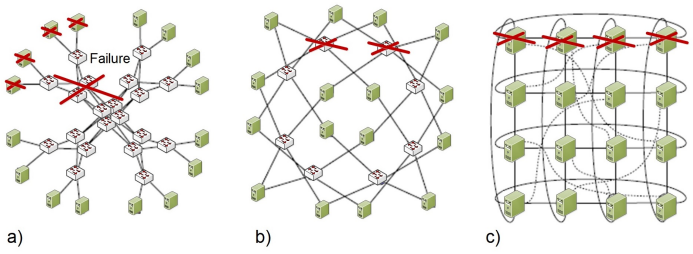

Fig. 7. Examples and comparison of network-centric vs. server-centric intra-DCN architectures [86], [87]. *a) Network-centric example:* FatTree, where a double switch failure disconnects 1/4 of the PMs. *b) Server-centric example:* BCube, where a double switch failure does not disconnect PMs. Note that even the failure of a third switch will only disconnect one PM. *c) A scalable and server-centric example:* SWDC Torus 2D, is the most resilient architecture in this figure, as even four (or more) switch failures do not disconnect the remaining network.

Server port redundancy design adds network connections and routing options in the servers (server-centric architectures) [117]. Some server-centric architectures enabling high port redundancy are *DCell* [82], *BCube* [83], *FiComm* [84], *DPillar* [85], *small-world data center (SWDC)* [86], and *Scafida* [87].

Table II
APPROACHES FOR RESILIENCY IN NETWORKS

Techniques and approaches					Contribution for cloud infrastructure resiliency / Type and level of resiliency enforced							
Type	Approach/es	Element	Resiliency technique	Method	Benefits	Challenges	Node	Link	Software*	Physical†	Attacks€	Disasters††
	Physical layer											
			Intra-DCN design									
DES	FatTree [79], VL2 [80]	Intra-DCN	Switch redundancy	RM/PR	High redundancy	Point of failure	Single	Single	-	++	+	-
DES	Qfabric [81]	Intra-DCN	Switch redundancy	RM/PR	High redundancy	High cost	Multiple	Multiple	-	+++	+	-
DES	Dcell [82], Bcube [83]	Intra-DCN	Server port redundancy	RM/PR	High redundancy	High complexity	Multiple	Multiple	-	++	+	-
DES	Ficom [84], Dpillar [85]	Intra-DCN	Server port redundancy	RM/PR	High redundancy	High complexity		Multiple	-	++	+	-
DES	SWDC [86]	Intra-DCN	Server port redundancy	RM/PR	High redundancy	High complexity	Multiple	Multiple	-	+++	+	-
DES	Scafida [87]	Intra-DCN	Server port redundancy	RM/PR	High flexibility	High complexity	Multiple	Multiple	-	+++	+	-
PLA	FScafida [88]	Intra-DCN	Topology augmentation	RM	High flexibility	High complexity	Multiple	Multiple	-	+++	+	-
			DCN traffic protection									
RUN	HADCN [89]	Intra-DCN	Traffic distribution	PR	Low traffic RT	Only for switches	Multiple	Multiple	+	++	+	-
RUN	BSR [83]	Intra-DCN	Route replication	PR	Low route RT	High overhead	Multiple	Multiple	+	++	-	-
RUN	ESR [88]	Intra-DCN	Route replication	PR	Low cost and RT	High Complexity	Multiple	Multiple	+	++	-	-
			DCN traffic recovery									
RUN	DFR [82]	Intra-DCN	Traffic recovery	RE	Low route RT	Works in Dcell	Multiple	Multiple	+	++	-	-
RUN	TAR [84]	Intra-DCN	Traffic recovery	RE	Low route RT	Works for Ficomm	Multiple	Multiple	+	++	-	-
			Inter-DCN traffic protection									
PLA	DBPP [90]	Inter-DCN	Pre-configured path	PR	Low traffic RT	Additional cost	Single	Multiple	-	+	-	+
PLA	SBPP [91]	Inter-DCN	Pre-configured path	PR	Low cost and RT	Multiple failures	Single	Single	-	+	-	+
PLA	p-Cycle [92], [93]	Inter-DCN	Pre-configure graph	PR	Low traffic RT	High cost	Single	Multiple	-	++	-	+
PLA	p-Tree [94]	Inter-DCN	Pre-configured graph	PR	Low cost and RT	Multiple failures	Single	Single	-	++	-	+
PLA	p-Prism [95]	Inter-DCN	Pre-configured graph	PR	Low traffic RT	Multi-tenancy	Multiple	Single	-	++	-	+
P/R	DR-Risk [96], [97]	Inter-DCN	Failure-aware provisioning	FR/RE	Disaster recovery	Risk determination	Multiple	Multiple	+	+++	+	+
	Virtualization layer											
			Survivable VN mapping									
RUN	RWA-Cut [98], [99]	VN	Topology cut avoidance	PR/RM	Low cost and RT	Multiple failures	Single	Multiple	-	++	-	-
RUN	SVTR [100]	VN	Topology cut avoidance	PR/RM	High survivability and low RT	Multiple domains	Multiple	Multiple	-	+++	+	+
P/R	Aug-SVON-map [101]	VON	Virtual topology augmentation	PR/RE	Partial re-provision	Complexity	Single	Multiple	+	++	+	-
RUN	Des-dim-SVT [102]	VN	Virtual topology augmentation	RE	High survivability and RT	Complexity	Multiple	Multiple	+	+++	+	-
P/R	Ext-aug-SVON [103]	VON	Traffic disruption avoidance	PR/RE	Low cost	Multiple domains	Single	Multiple	+	++	+	+
RUN	Rout-alloc-SVON [104]	VON	Traffic disruption avoidance	PR/RE	Low cost	Multiple failures	Single	Single	+	++	+	-
			Cross-layer protection									
P/R	CL-SVNE[105]	VN	Physical-layer backup sharing	PR/RE	Low cost	Multiple failures	Single	Single	+	+	-	+
RUN	SBPSV [106]	VN	Physical-layer backup sharing	PR/RE	Low cost	Multiple failures	Single	Single	+	+	+	-
P/R	SBPSVO [107]	VON	Physical-layer backup sharing	PR/RE	Low cost	Multiple failures	Single	Single	+	+	+	-
RUN	SPGC [108]	VN	Two-layer backup optimization	PR/RE	High survivability and low cost	Multiple domains	Multiple	Multiple	+	+	++	+
			Service-oriented VN recovery									
RUN	QoS-VNmap [109]	VN	SLA-aware survivability	PR/RE	Low RT and cost	Multiple domains	Single	Single	+	+	+	-
P/R	OVN [110], HOVN [111]	VON	SLA-aware survivability	PR/RE	Low RT and cost	Scalability	Single	Single	+	+	+	+
RUN	VPCS [112]	VN	SLA-aware survivability	PR/RE	Adaptability	Multiple domains	Single	Single	+	+	+	+
			Resiliency for software-defined network									
RUN	CP Recovery [113]	SDN	Controller replication	PR	High survivability and Low RT	Multiple domains	Multiple	Multiple	+	++	+	-
RUN	SDN-DR-WAN [114]	SDN	Controller placement	RM/PR	Disaster Recovery	High complexity	Multiple	Multiple	+	+++	+	+

Techniques and approaches
Type, Method: as in Table I.
Level of resiliency
*Survivability to different extent of software failures, "+" failure of some routes or traffic flows.
†Survivability to different extent of hardware failures, "+" failure of a single link or node (switches and/or routers and/or servers), "++" failure of some links and nodes, "+++" failure of several links and nodes.
€Survivability to different extent of attacks, "+" attacks with easy recovery.
††Survivability to different extent of disasters, "+" disaster without cascading failures affecting a set of links and nodes.

Figures 7(b) and 7(c) present examples of two server-centric architectures (SWDC and BCube) that protect against multiple failures in servers and switches (SWDC is the most resilient proposal) (Table II).

Topology augmentation design technique is the assisted addition of network links and/or nodes to improve the resiliency of Intra-DCN architectures. One approach using this technique is the *full-fledged DC architecture (FScafida)* [88] to guide topology augmentation in the Scafida architecture [87].

1.b) *Intra-DCN traffic protection:* consists in the distribution of traffic and/or replication of routes for case of link and/or node failures.

Traffic distribution: consists in the traffic splitting, distribution, routing, and load balancing through different nodes and links to avoid losses due to failures. One such approach for redundant switch architectures (i.e., Fattree and VL2) is the *high-availability and full bandwidth communication (HADCN)* suggested in [89]. The main advantage of this approach is the adaptability and the small additional capacity required to provide resiliency (Table II).

Route replication: consists in the use of multiple routes (i.e., k-shortest paths) to provide protection up to k-1 links and/or node failures. An approach using this technique is the *BCube source routing protocol (BSR)* proposed in [83], however the main challenge of this approach is the high overhead produced in the addition of a new requested route. To deal with this weakness, *effective source routing (ESR)* is suggested in [88] to decrease the cost of route replication compared to BSR (see Table II).

1.c) *Inter-DCN traffic protection:* consists in the pre-provisioning of redundant capacity to protect network traffic in case of failure. The topic of protection in generic communication (transport) networks has been largely investigated (see, e.g., [118], [119]). In summary, three types of approaches using this strategy are as follows:

Pre-configured path: techniques that pre-provision backup paths to protect traffic on the network. Two types of pre-configured approaches are used in inter-DCN; *dedicated backup path protection (DBPP)* [90], which requires a dedicated backup for each working path; *shared backup path protection (SBPP)* [91], which allows sharing of backup paths between different working paths. Both approaches offer full protection for each working path against link and node failures. DBPP requires more cost in resources compared to SBPP, however DBPP might provide high resiliency for case of multiple failures.

Pre-configured graph: consists in pre-provisioning of redundant capacity using graph topologies. Some approaches enabling this techniques are *pre-configured tree (p-Tree)* [94] based on tree graphs, *pre-configured cycles (p-Cycle)* [92] based on cycle graphs, and *pre-configured prism (p-Prism)* [95] based on prism graphs. p-Cycle can be combined with pre-configured path (i.e., FIPP-p-Cycle and flow p-Cycle) [92], [93]. Figure 8 shows an example of FIPP-p-Cycles scheme used in a large and distributed cloud infrastructure (i.e., Inter-DCN) protecting traffic and connections from a double failure in the core or transmission network.

Failure-aware provisioning: consists in avoiding potential failures using some predictions (i.e., probability of disaster occurrence) to reduce the risk of cloud service disruptions. Risk-aware provisioning techniques for protection and recovery routes and paths in multiple link and node failures caused by a disaster is *disaster resilient risk aware (DR-Risk)*, as suggested in [96], [97].

1.d) *DCN traffic recovery:* consists in fast traffic re-routing in case of failure in network components. One approach enabling traffic re-routing on each intermediate node or server in case of Intra-DCN failures is *traffic-aware routing (TAR)* [84]. A similar approach for the DCell architecture is the *DCell fault-tolerance routing (DFR)* [82]. Both approaches work on a best-effort basis, hence re-routing in case of multiple failures or disaster can be challenging (Table II). Various approaches for inter-DCN traffic recovery are surveyed in [118].

Note that some approaches for *Intra-DCN reliable design* were proposed by network hardware industry (e.g., Qfabric from Juniper) to address the requirement for redundant ports and specialized switches. The remaining approaches are mostly proposed by academia.

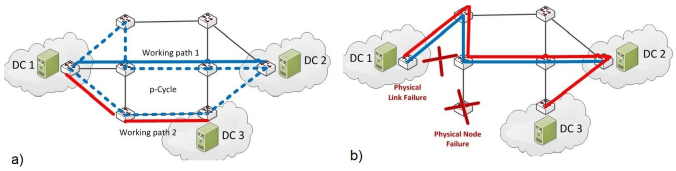

Fig. 8. Example of a p-Cycle. a) FIPP p-Cycle (dotted line) used to protect two working paths. b) Example of double failures and the use of p-Cycle for recovery.

2. *Resiliency in virtualized networks*

We classify three approaches to network virtualization: virtual optical network (VON), virtual private network (VPN), and software-defined network (SDN). VON uses hardware-level virtualization (Fig. 9) where the mapping of each virtual link uses optical channels, while VPN and SDN are based on software-level virtualization. VPN is a type of VN created by the generalized multiprotocol label switching (GMPLS) or tunneling protocols [120]. SDN introduces programmability and independence between control and data planes [121], [122].

The major challenge of resiliency in virtualized networks is that a single physical failure might disconnect and affect many virtual networks. To deal with this problem, survivable virtual network embedding (SVNE) approaches were suggested in [116], [123]. Figure 9 shows an example of the SVNE problem and some techniques to deal with failures.

Surveyed approaches for resiliency in virtualized networks follow the classification in Fig. 6 and are summarized in Table II. Note that these approaches can be either proactive (redundant capacity assignment happens before the failure)

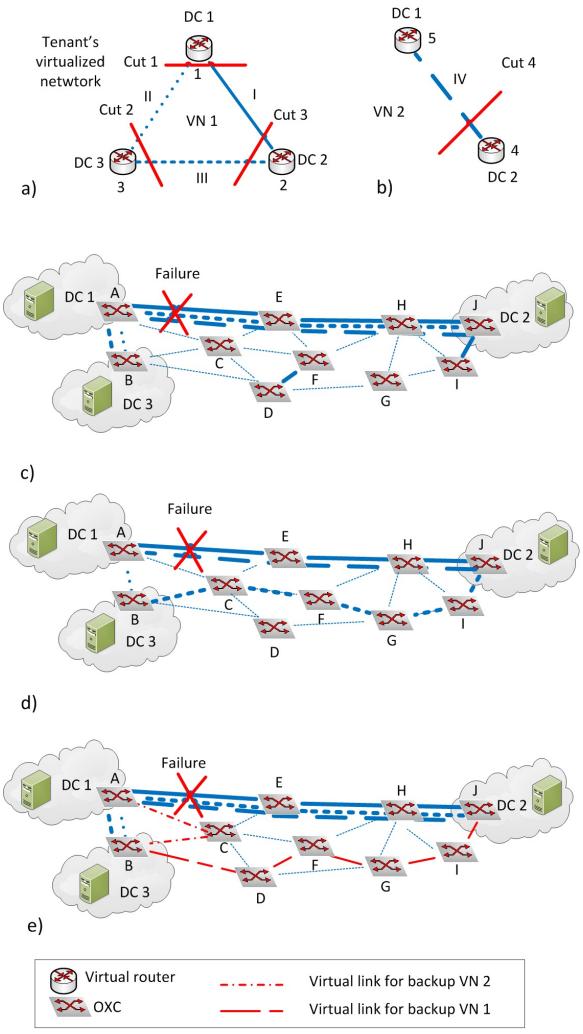

Fig. 9. Example of SVNE problem and two suggested techniques. a-b) Two virtual networks VN 1 = [1-2, 2-3, 1-3] (virtual links I, II, and III) and VN 2 = [5-4] (virtual link IV) connecting tenants in a federated cloud comprising three data centers [DC 1, DC 2, and DC 3]. The virtual topology of VN 1 has three cuts while the virtual topology of VN 2 has one cut. c) Example of non-survivable mapping of both VNs using one end-to-end optical channel (lightpath) per virtual link. Any failure in one or in a combination of the optical links [A-B, A-E, E-H, H-J] will disconnect VN 1 and VN 2 because one or more cuts share the same optical links (e.g., both cuts 3 and 4 are disconnected if physical link A-E fails). d) Survivable (i.e., cut-disjoint) mapping, where no single link failure disconnects VN 1 (VN 2 cannot be protected, as it it composed by a single virtual link). e) Topology augmentation technique where an additional backup virtual link is mapped for each non-survivable VN to increase their resiliency.

or reactive (backup capacity is provisioned after the failure). Most techniques provide resiliency in single-link or single-node failure scenarios except resilient cut-disjoint, cross-layer backup, and SDN resiliency techniques, which provide higher resiliency in case of multiple nodes or disaster failures.

2.a) *Survivable VN mapping:* consists in provisioning capacity for the mapping of a virtual network over a physical network in such a way that a failure (or a set of failures) at the physical layer do not disconnect the virtual network. We classify three types of approaches for survivable VN mapping: topology cut avoidance, virtual topology augmentation, and traffic disruption avoidance (see Table II).

Topology cut avoidance: consists in performing VN mapping by avoiding that virtual links belonging to the same cut in the virtual topology are mapped on the same physical link (otherwise the failure would disconnect the virtual network). One of the earliest proposals of cut-disjoint mapping for survivable virtual optical network (VON) mapping in IP-over-WDM networks is an extension of the routing and wavelength assignment (RWA) called *RWA-cut-disjoint* [98], [99]. Cut-disjoint mapping technique offers a more scalable resiliency compared to path-disjoint approaches introduced in other studies such as [124]. Given that RWA-cut-disjoint enforces near-full VON survivability against single-physical-link failures and a best-effort resiliency for dual-link failures, the *survivable virtual topology routing (SVTR)* is also suggested in [100] to deal with larger failures (i.e., a group of links and/or nodes which might fail together, a so-called shared-risk group (SRG)). An example of cut-disjoint mapping is presented in Fig. 9(d).

Virtual topology augmentation: requires the addition of virtual links and/or nodes to protect or recover possible failures. One of the earliest VN mapping approaches using this technique is *virtual network topology augmentation for survivable VON mapping (aug-SVON-mapping)* [101] where a non-survivable VON mapping is augmented to become survivable. Aug-SVON-mapping identifies if cuts of the VN topology can be disconnected by a single-link failure and then adds a virtual link in order to avoid the disconnection. Multiple augmentations can provide protection or recovery against multiple link failures. The *design-dimensioning survivable virtual topology (des-dim-SVT)* approach in [102] reduces considerably the RT and allows multiple topology augmentation for one and multiple VNs or multi-tenants VNs, which is key for cloud infrastructure providers. An example of a single-link topology augmentation is presented in Fig. 9(e) where one virtual link is mapped in addition to each VN to convert the non-survivable VN mapping of Fig. 9(c) into a survivable VN.

Traffic disruption avoidance: consists in mapping the VN to avoid specific routes and/or traffic disruptions that can be know a priori. Two such approaches are *ext-aug-SVON* [103] that uses failure-aware routing in the topology augmentation scheme, and *routing-allocation-SVON* in [104] that also attempts to limit congestion while performing failure-aware routing. Ext-aug-SVON also provides post-failure fast-recovery capabilities, similar to the recovery schemes discussed previously. In the case of routing-allocation-SVON, the routing of IP-level and optical-level paths are jointly optimized to minimize the required capacity. Hence, routing-allocation-SVON provides higher cost-efficiency, while ext-aug-SVON might provide higher resiliency as it might recover from a larger number of failures.

2.b) *Cross-layer VN protection:* groups the approaches where protection (or restoration) of the virtual networks is provided through a cross-layer design of backup capacity at

the physical and logical (virtual) network levels. We subdivide this into two categories, one focusing on using physical-layer backup capacity sharing, and the other focusing on two-layer backup paths optimization (Table II).

Physical-layer backup sharing: focuses on the effective use of backup capacity in the physical network. The *cross-layer survivable virtual network embedding (CL-SVNE)* proposed in [105] is a hybrid approach that provides both proactive and reactive survivable VN mapping by combining two-link physical protection and VN mapping. However, CL-SVNE might require a large amount of backup capacity in the physical layer. Hence, backup capacity sharing between VNs can be enabled using the *shared backup network provision for SVN (SBPSV)* in [106] or *SBPSV for SVON (SBPSVO)* in [107] to provide a more capacity-efficient survivable VN mapping.

Two-layer backup optimization: consists in the interchange of backup resources between the physical network and the virtual network. This technique extends the previous cross-layer protection schemes to provide resiliency in case of larger failures (i.e., SRG and disasters). One such approach is the *inter-layer shared backup path (SPGC)* in [108] that exploits a two-layer shared backup pool (similar to a generalization of the common pool technique in MPLS networks), where the shared backup pool comprises either standby virtual or physical links that can be used to recover different kinds of failures.

2.c) *Service-oriented VN recovery:* consists in performing VN recovery while explicitly accounting for service-level objectives (SLO) that can be different for different tenants. One service-oriented VM recovery approach is *QoS-VNmap* [109] that uses SLO requirements to allocate the backup capacity needed to meet the specific RT of each cloud application. A similar approach is *overlay VN mapping (OVN)* in [110]. Since OVN is not scalable for large multi-tenant cloud infrastructure, the *heuristic overlay VN mapping (HOVN)* approach in [111] has also been proposed to add scalability while preserving the SLO targets and quasi-optimality of the results. Finally, for the specific case of MPLS VPN services (which reserve capacity in each router or switch based on QoS requirements), the approach *VN configurable survivability (VPCS)* is proposed in [112] to add traffic forwarding intelligence in each router and enable adaptable protection and recovery for the cases of node or link failures.

2.d) *Resiliency for software-defined networks (SDN).* To enable virtual networks, the interaction between data plane and control plane is key for resiliency. Some studies propose to adopt software-defined networking (SDN) to deal with this problem [121]. Resiliency in SDN networks requires replication of the centralized controller and careful placement of these replicas. The *CP Recovery* approach in [113] proposes a scheme for the physical replication of the logically-centralized controller while minimizing RT. *SDN-DR-WAN* in [114] extends the replication scheme in [113] by adding a disaster-resilient placement of multiple controllers in a large network. Experimental results confirm the applicability of SDN-DR-WAN for disaster recovery (see Table II).

Discussion of benefits and challenges for resiliency techniques in networks: Table II presents key benefits and challenges of resiliency in networks. Some benefits for cloud infrastructure resiliency are: (i) High redundancy provided by techniques for Intra-DCN design; (ii) Low cost by techniques for cross-layer protection and survivable VN mapping because they allow cooperative resource allocation between physical and virtualized networks; (iii) Low recovery times (RT) provided by reserving additional capacity (i.e., Inter-DCN protection techniques), and delay is minimized by service-oriented VN recovery; (iv) Flexibility and multi-tenancy capabilities are obtained by resiliency techniques for network virtualization. However, resiliency techniques in networks face similar challenges of the resiliency techniques in servers, especially in terms of complexity and the necessity of integration with other techniques to provide resiliency in cloud infrastructure. The mentioned challenges are discussed in following Sections V-D, V-E, and V-F).

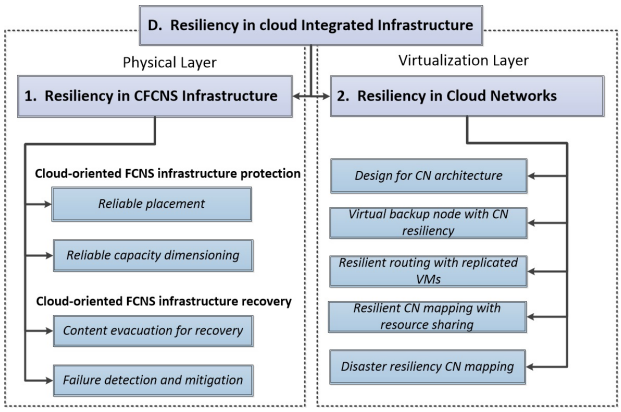

Fig. 10. Cloud integrated infrastructure taxonomy.

D. Resiliency in cloud integrated infrastructure

Cloud providers connect computation and communication elements in an integrated infrastructure to offer their services to tenants and customers. This integration results in a so-called Cloud Oriented Federated Computing and Networking System (CFCNS) [157]. Then, cloud functionalities (as seen in Section II-B) are enabled by virtualizing the CFCNS infrastructure, i.e., slicing the CFNS resource, thanks to virtualized servers (i.e., VMs and VSs) that are accessed using virtualized networks (i.e., VN, VON, and SDN). In the following, we will call the integration of virtual servers and virtual networks as a Cloud Network (CN) [155], i.e., the CN is a virtualized slice of the CFCNS infrastructure (see Glossary in Section VIII). In such an integrated system, failures in network elements as well as in servers might induce failure cascades that affect other elements of the CFCNS, given their strict interdependence. So, resiliency techniques specifically designed for communication network or server systems in isolation can be insufficient to ensure the resiliency of a CFCNS. An efficient coordination and integration of these resiliency approaches is necessary.

Table III
APPROACHES FOR RESILIENCY IN CLOUD INTEGRATED INFRASTRUCTURE

Techniques and approaches					Contribution for cloud infrastructure resiliency / Type and level of resiliency enforced							
Type	Approach/es	Elements	Resiliency strategies	Methods	Main Benefit	Main Challenge	DC	Network	Software*	Physical†	Attack€	Disaster††
	Physical layer											
			Reliable placement									
DES	AHP [125]	C/R/DC	VA/Place	RM/PR	High survivability	Complex selection	Single	Single	-	++	-	-
PLA	DRDCP [126], [127]	C/R/DC	VA/DR/Place	RM/PR	High survivability	Multiple domains	Multiple	Multiple	++	+++	++	+
			Reliable capacity dimensioning									
RUN	R-Aware [128]	C/R/DC	CD/AnyCast/Place	PR	Low cost	Adaptability	Single	Multiple	+	+++	+	-
PLA	DAP [129]	C/R/DC	CD/DR/AnyCast/Place	PR	Low cost	High complexity	Single	Multiple	+	++	++	+
PLA	FID [130], [131]	C/R/DC	CD/AnyCast/Place	RE	Adaptability	Multiple domains	Single	Multiple	+	++	+	-
PLA	DRDD [132]	C/R/DC	CD/DR/AnyCast/Place	PR	High survivability	Multiple domains	Multiple	Multiple	++	+++	++	+
			Content evacuation for recovery									
PLA	DB-Multi-DB [133]	C/R/DC	Evacuation/DR	PR	Low RT	Multiple domains	Multiple	Multiple	++	+++	++	+
RU/P	DR-RAPID-EVA [134]	C/R/DC	Evacuation/DR	RE	High survivability and low RT	Multiple domains	Multiple	Multiple	++	+++	++	+
			Failure detection and mitigation									
RUN	NetPilot [24]	C/R/DC	Failure detection/reset	FR/RE	Adaptability	Multiple domains	Single	Multiple	++	++	+	+
RUN	PFMLearning [135]	C/R/DC	Failure detection	FR	Adaptability	Only detection	Multiple	Multiple	+	++	+	+
	Virtualization layer											
			Resilient CN architecture									
DES	VICTOR [136]	DC/CN	CR/AM/VM survivability	RE/FR	Support CN adaptability	Non VN adaptability	Multiple	Multiple	-	++	+	+
DES	SecondNet [136]	DC/CN	CD/VM survivability	RE/FR	Support VM migration	Non VN adaptability	Multiple	Single	-	++	+	+
DES	Octopus [136]	DC/CN	VN survivability	RE/FR	Support CN adaptability	Non VM migration	Single	Multiple	-	++	+	-
DES	CloudNaaS [136]	DC/CN	AM/VM survivability/VN survivability	RE/PR	Support CN adaptability	Non CR	Multiple	Multiple	-	++	+	+
DES	VENICE [136]	DC/CN	CR/VM survivability/VN survivability	RE/PR	Support CN adaptability	Non AM	Multiple	Multiple	-	++	+	+
			Virtual backup node for CN resiliency									
PLA	l-Backup-Node [137]	VN/CN	VBN/VN survivability	PR	High survivability	Multiple domains	Single	Multiple	++	++	+	+
PLA	K-Backup-Node [137]	VN/CN	VBN/VN survivability	PR	High survivability	High cost	Multiple	Multiple	++	++	+	+
PLA	LSNE [138]	CN	VBN/VN survivability	PR	High survivability	Multiple domains	Single	Multiple	++	++	+	-
PLA	FD-K-Backup-Node [139]	CN	VBN/VA/VN survivability	RM/PR	High survivability	High cost	Multiple	Multiple	++	++	+	+
PLA	RSVIMB-Node [140]	DC/CN	VBN/VA/VN survivability	PR	High survivability	Multiple domains	Multiple	Multiple	-	++	+	+
R/P	SVI-Act-VM-Rep [141]	VM/CN	VBN/VM place	RE/PR	Low cost	Replica synchronization	Single	Multiple	+++	++	+	+
RUN	VN-GRASP [142]	VM/CN	VBN/VM replication	PR	Low cost	Multiple domains	Single	Multiple	+++	++	+	+
RUN	VNsnap [143]	VM/CN	CN checkpoint	PR	High survivability	Replica synchronization	Multiple	Multiple	+++	++	+	+
			Resilient routing with replication VMs									
RUN	SCN-AnyCast [144]	R/CN	AnyCast/VM replication	RE/PR	Lost cost	Replica synchronization	Multiple	Multiple	+++	++	+	+
RUN	SCN-Joint-Any-Rep [145]	R/CN	AnyCast/VM replication	RE/PR	Replica synchronization	Multiple domains	Multiple	Multiple	+++	++	+	+
RUN	RA-VDC-Emb [146]	R/CN	AnyCast/VM replication	RE/PR	Low cost	Multiple domains	Multiple	Multiple	+++	++	+	+
			Resilient CN mapping with resource sharing									
RUN	LP-RVIM [147]	CN	Backup sharing	RE/PR	Resource optimization	Multiple domains	Multiple	Multiple	++	++	+	-
R/P	SVI-ORP [148]	CN	Backup sharing	RE/PR	Adaptability	High complexity	Multiple	Multiple	++	++	+	-
R/P	SVI-REOP [149]	CN	Backup sharing	RE/PR	Adaptability	High complexity	Multiple	Multiple	++	++	+	-
			Disaster resilient CN mapping									
RU/P	FD-RRVIM [150]	VM/CN	VBM/VM-Check/DR	RE/PR	Low RT	Cascading failures	Multiple	Multiple	+++	+++	+	+
RUN	DS-AnyCast-BNode [151]	R/CN	VBM/AnyCast/DR	RE/PR	Adaptability	Cascading failures	Multiple	Multiple	+++	+++	+	+
RUN	RVN-DR-Content [152]	C/R/CN	Place/AnyCast/DR	RE/PR	Adaptability	Multiple domains	Multiple	Multiple	+++	+++	+	+
PLA	FRGBM [153]	CN	VA/DR	FR/RM	Low cost	Cascading failures	Multiple	Multiple	+++	+++	+	+
R/P	DFRDM [154]	CN	VA/DR	FR/RM	Low cost	Cascading failures	Multiple	Multiple	+++	++	+	+
PLA	DRM-PDS [155], [156]	CN	VBN/VA/Cut/DR	RE/PR	Cascading failures	High complexity	Multiple	Multiple	++	++	++	++
RUN	Pipe-Cloud [30]	CN	VM survivability/DR	RE/PR	Low RT	Multiple domains	Single	Multiple	+++	++	+	+
RUN	Second-Site [21]	CN	Failure Detection/VM migration/DR	RE/PR	Fast recovery	Multiple domains	Single	Multiple	+++	++	+	+

Type of approach "DES" design of cloud infrastructure, "RUN" run-time, "PLA" planning of capacity, and "R/P" Run-time or planning or both.
Element used by the approach, "C" content or cloud computing data, "R" routing or provisioning for communication, "DC" entire data center, "CN" whole cloud network, "VN" virtual network, "VM" virtual machine.
Strategies used by the approach,
"VA" vulnerability aware, "CD" capacity dimensioning, "Place" content/DC placement, "AnyCast" routing, "DR" disaster resiliency, "VBN" virtual backup node mapping, "CR" central controller replication functionality, "AM" address mobility functionality, "VM survivability" general VM survivability functionality, which can be any or combination of "VM migration" virtual machine migration, "VM replication" virtual machine replication, "VM chekpointing" virtual machine checkpointing, "VN survivability" VN survivability functionality, which can be "Cut" Cut-disjoint, "PC" pre-configured capacity.
Methods "PR" is protection, "RE" recovery, "FR" forecasting, "RM" removal, "RE/PR" either recovery or protection or both.
Level of resiliency enforced
*Survivability to different extent of cloud application failures, "+" minor failure in process and data, "++" failure of multiple processes and data (i.e., cloud services temporarily unreachable) , "+++" failure in the whole application (i.e., cloud services corrupted and with losses).
†Survivability to different extent of cloud infrastructure failures, "+" failure of single DC component, e.g., one server or a communication device in the DCN, "++" failure of one entire DC and/or entire link of the inter-DCN, "+++" failure of many links and components of the inter-DCN, and/or more than one DC is disconnected.
€Survivability to different extent of attacks, "+" attacks on one tenant and minor services disruption, "++" attacks affecting a DC with data losses for more than one tenant (difficult to mitigate).
††Survivability in case of disaster and post-disaster cascading failures, "+" one disaster with low risk of cascading failure, "++" more severe disaster with moderate risk of cascading failures.

In continuation to the previous classifications, we categorize approaches for CFCNS resiliency as those related to: (i) the physical layer or (ii) the virtualization layer. Surveyed approaches for resiliency in cloud integrated infrastructures are organized according to the classification in Fig. 10, while Table III introduces a taxonomy of the approaches, followed by a brief description of each approach and technique.

1. *Resiliency in CFCNS infrastructures:* covers those approaches for protection and recovery of content, routes, and DCs that apply in the case of failures affecting the CFCNS (i.e., not addressing the specific issue of virtualization as the next category). The four main techniques are:

1.a) *Reliable placement:* which consists in performing, during the design phase, reliable placement of DCs (or VMs) and reliable choice of routes in a CFCNS infrastructure. Two surveyed approaches fall in this category: *analytic hierarchy process (AHP)* in [125] selects the most survivable placement for the nodes and links of the inter-DCN network topology based on the vulnerability of physical links (i.e., vulnerability aware (VA)); the *disaster resiliency data center and content placement (DRDCP)* in [126], [127] minimizes the risk of damage produced by a disaster (i.e., natural disasters and WMD attacks) on cloud facilities and contents.

1.b) *Reliable capacity dimensioning:* adapts and organizes the capacity of CFCSN to avoid failures. Existing approaches for capacity dimensioning typically rely on anycast routing and resource degradation. Anycast routing can be used to discover the minimum-cost connectivity between all possible pairs of DCs or contents. Resource degradation is a temporary reduction of the server or bandwidth capacity enforced to distribute surviving resources to all customers. Three approaches using one or both these techniques are: *routing aware placement (R-Aware)* [128], *failure independent rerouting (FID)* [130], [131], and *degradation-aware provisioning disaster recovery (DAP)* [129]. The three approaches minimize cost and RT by using: connection relocation (in FID), two types of routing protection (in R-Aware), and two types of backup routing with gradual degradation in case of multiple failures (in DAP). An extension of DAP for gradual capacity degradation to recover connections after a disaster is suggested in [158]. *Disaster resilient data center dimensioning (DRDD)* in [132] also adds content replication and placement in the capacity dimensioning.

1.c) *Content evacuation for recovery.* In presence of predictable failures (e.g., an upcoming hurricane), an alert is typically issued and the cloud provider might have time to "evacuate" data from the data center locations under risk towards other safe data center facilities. Two approaches implementing this concept are *disaster backup in multi-data center (DB-multi-DC)* [133] and *rapid data-evacuation strategy (DR-RAPID-EVA)* [134]. DB-multi-DC uses a two-step algorithm to solve the fast-backup evacuation of data center contents. DR-RAPID-EVA exploits the time made available by the disaster alert and the extent of the forecast damage as input to run a fast reactive evacuation algorithm, especially tailored for large failures such as disasters or Electromagnetic Pulse (EMP) attacks.

1.d) *Failure detection and mitigation:* consists in a set of resiliency techniques that support failure detection, mitigation, and recovery in cloud infrastructure. A well-known approach is the *automatic data center network failure mitigation (NetPilot)* in [24] that implements failure detection by inferring device anomalies, then disables the failed components, and migrates cloud services to failure-free components until the failed component is repaired or replaced. A similar approach, *Failure managed by integrated unsupervised and semi-supervised learning (PFMLearning)* [135] introduces a failure-detection approach that uses Bayesian classifiers to detect and classify anomalous behaviors (i.e., possible software and hardware failures), followed by an application of learning decision trees to process such anomalies and possibly to predict incoming failures in large cloud infrastructures. This approach combines measurement and estimation technique (Section V-F) with runtime failure mitigation.

2. *Resiliency in cloud networks:* covers the resiliency approaches working at the virtualization layer, i.e., approaches exploiting the concepts of virtual servers, virtual networks (and their integration), and the cloud network (CN). Various techniques, comprising the virtual backup node (VBN), VM migration, VM replication, CN mapping, anycast routing, and backup sharing are overviewed in this section. Figure 11 presents an example of a CFCSN architecture supporting a CN comprising VNs and VMs (see Glossary in Section VIII).

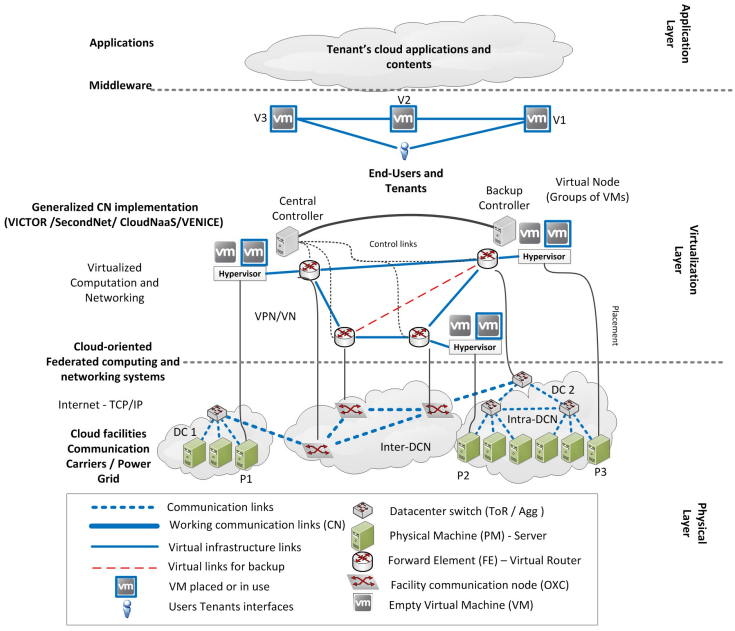

Fig. 11. Three layers of a CFCSN architecture and its components at each layer. A generalized CN is instantiated on top of the CFCNS infrastructure based on two fat-tree intra-DCNs. This generic example is applicable to any of the discussed resilient architectures: VICTOR, SecondNet, CloudNaaS, and VENICE. A replicated centralized controller is also supported.

2.a) *Resilient CN architectures:* contain a set of studies

proposing functionalities to support resilient CN design. A survey of CN architecture is presented in [136], with proposals for resilient CN mapping and design. Most surveyed approaches rely on combinations of the following families of functionalities: (i) *central controller replication (CR)*, (ii) *redundant (bandwidth and processing) capacity dimensioning*, (iii) *IP and/or applications address mobility (AM)*, (iv) *VM survivability functionalities (VM survivavility)* (e.g., VM migration, VM replication, and VM placement), (v) *VN survivability strategies (VN survivaviliy)* (e.g., pre-configured backup paths, cut-disjoint mapping, anycast routing, etc.). As shown in Table III, [136] surveyed five main proposals in this category: VICTOR, VENICE [146], and Oktopus (network-centric Intra-DCN architectures), CloudNaaS, and SecondNet (server-centric architectures). Figure 11 illustrates a generic example of a CN mapped over a CFCNS infrastructure, applicable to all the previous five approaches. Another earlier CN implementation is the virtual inter-networking on overlay infrastructure (VIOLIN) suggested in [159].

2.b) *Virtual backup node for CN resiliency:* consists in the utilization of an external virtual backup node (VBN) to be placed in an additional node or DC to provide CN protection and recovery in case of failure of one of the original VN nodes. Each backup facility node is connected with additional virtual links and provides reserved capacity to replicate and migrate VMs in case of failures. Application of the VBN technique requires effective selection and/or placement of the VBN. Two types of VBN placement strategies were suggested: (i) failure independent, as in *l-Backup-Node and K-Backup Node* [137] and *location constrained survivable network embedding (LSNE)* [138]; and (ii) failure dependent, as in *failure-dependent backup node approach (FD-K-Backup-Node)* [139], [137], and *survivable virtual infrastructure mapping for regional failure in FCNS infrastructure (RSVIM-Bnode)* [140]. l-Backup-Node consists in the mapping of one VBN per CN while k-Backup-Node consists in the mapping of one VBN for each virtual node. LSNE is similar to K-Backup-Node, while FD-K-Backup-Node and RSVIN-Bnode map the VBN based on the level of resiliency required. RSVIN-Bnode's main goal is to map CNs protected from disaster failure. Figure 12 introduces an example of a CN using VBN technique. The VBN technique is also often used in combination with other techniques such as VM replication, VM placement, and VM checkpoint techniques (that we have introduced before). VBN with VM replication approaches are *survivable virtual infrastructure with active VM replication (SVI-Active-VM-Rep)* [141] and *a greedy randomized adaptive VM replication (VN-GRASP)* [142]. VBN with VM checkpoint is suggested in *VNsnap* by [143]. SVI-Active-VM-Rep divides the problem in two sub-problems, VM placement and survivable virtual link mapping, and solves both using a heuristic algorithm which minimizes the reserved bandwidth. VM-GRASP optimizes resource allocation including the future connectivity demands, and requires less resources compared to SVI-Active-VM-Rep. VNsnap stores snapshots of the entire CN, including processing, storage, and communication states, which can then be recovered in selected backup facility nodes. VNsnap is more cost-efficient compared to VM-GRASP and SVI-Active-VM-Rep (Table III).

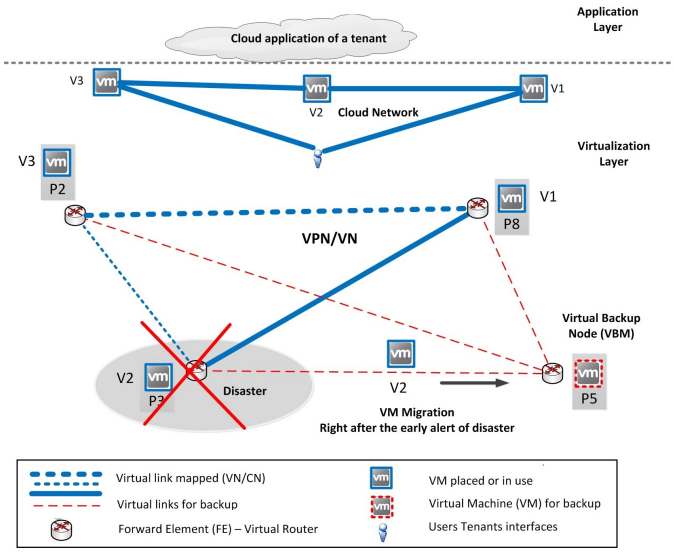

Fig. 12. Example of CN resiliency using the Virtual Backup Node. The CN connects three VMs using a VN. The virtual backup node is instantiated using VM migration in case of failure.

2.c) *Resilient routing with replicated VMs:* uses anycast routing towards replicated VMs to minimize delay, cost, and resources required. *Survivable CN with anycast (SCN-AnyCast)* [144] minimizes delay and/or cost in CN mapping with replicated VMs in case of DC and link failures. However, the synchronization between VM replicas requires additional mapping, hence [145], [160] propose an extension of SCN-AnyCast, called (*SCN-Joint-AnyCast-Rep*), that performs joint optimization of synchronization bandwidth while minimizing RT. A more comprehensive and cost-efficient approach using anycast routing with VM replication is the *reliability-aware VDC embeding (RA-VDC-Emb)* approach [146] that maps, replicates, and migrates VMs in case of server or communication failures.

2.d) *Resilient CN mapping with resource sharing techniques:* consists in the sharing of facility nodes (i.e., server and/or DC), and/or link capacity as pool of resources (i.e., resource pooling) between different CNs to provide resiliency. Approaches using sharing provide cost-efficient protection and recovery. *Local protection for resilient VI mapping (LP-RVIM)* [147] is a node/link sharing approach that uses a node and links already mapped by existing CNs as virtual backup nodes for new incoming CN requests. Resource pooling is enabled also in *virtual infrastructure mapping with opportunistic resource pooling (SVI-ORP)* [148] to enforce resiliency with "k-redundant shared resources" placed in strategic locations. SVI-ORP extends the concept of SBPSV [106] for CN resiliency. However, resource pooling can produce resource fragmentation which is very difficult to optimize in large and multi-domain cloud infrastructure. *Survivable virtual infrastructure*

with resource optimization (SVI-REOP) in [149] uses an auxiliary graph transformation to minimize resource fragmentation.

2.e) *Disaster resilient CN mapping* covers a set of approaches combining and extending the previously-discussed resiliency techniques to provide protection and recovery in case of disaster failures. Note that, in disaster scenarios, the initial failures directly caused by the disaster are usually followed by *correlated and cascaded post-disaster failures* (due to, e.g., aftershocks, or power shortage due to battery exhaustion). Different approaches for different types of disaster failures are summarized in Table III. The first approach introduced in Table III is the *regional failure-independent resilient mapping (FD-RRVIM)* [150] that extends RSVIM-Bnode by adding VM checkpointing, replication, and migration into VBN technique to provide an adaptive, cost-efficient, and fast CN recovery in case of a disaster. However, VBN requires additional cost for CN mapping which might not be sustainable, especially in a multi-tenant and multi-domain cloud environment. Hence, *disaster survivable anycast (DS-AnyCast-BNode)* [151] uses anycast routing to reduce the capacity required to connect and place VBN and VM replicas. A similar approach, *disaster resilient mapping with content connectivity (RVN-DR-Content)* [152], maps CNs to provide disaster-resilient content placement without using a specific backup node to reduce cost and complexity.

In [152], the concept of network connectivity (ensuring reachability among all nodes of the CN) is evolved to content connectivity (ensuring reachability of content replica, even if the CN is disconnected) for the case of disaster failures. Two other approaches, *failure region disjoint mapping (FRGBM)* [153] and *dynamic failure region disjoint mapping (DFRDM)* [154], use disaster-aware (i.e., disaster-disjoint) mapping to add backup virtual links where failures may occur under the objective of minimizing the overall cost (Table III). Most of the surveyed approaches use the a priori knowledge of disaster probability (determined using hazard maps [97]) to map a CN by avoiding disaster-prone areas. Nonetheless, most disaster-resilient CN mapping schemes do not consider post-disaster correlated and cascading failures. *Disaster resilient mapping with post-disaster survivability (DRM-PDS)* [155], [156] adds post-disaster-risk awareness into VBM and cut-disjoint routing to minimize the risk of CN disconnection and capacity losses caused by post-disaster cascading failures. Figure 12 shows an example of disaster-resilient CN mapping using VBN and VM migration.

Finally, two disaster-recovery approaches applying a complete replica of entire cloud sites are *cloud based disaster recovery (PipeCloud)* [30] and *disaster tolerance SecondSite* [21]. PipeCloud replicates and synchronizes entire cloud sites or cluster of VMs of multiple CNs through large cloud computing infrastructure to create recovery capability, while SecondSite detects and recovers cloud sites in multi-domain clouds affected by a disaster (see Table III).

Discussion of benefits and challenges of resiliency techniques in cloud integrated infrastructure: Table III compares approaches on resiliency in cloud infrastructures and discusses some benefits and challenges. The main benefits for cloud infrastructure designer and providers are: (i) integration of server and network resiliency planning and operation; (ii) virtualization, which adds cost efficiency and flexibility in multi-domain clouds; (iii) comprehensive failure and damage estimation tools based on risk assessment (e.g., R-Aware, DRDD, DRM-PDS), machine learning, and probabilistic tools (e.g., Net-Pilot, PFMLearning); and (iv) resiliency planning (i.e., PLA) and operation (i.e., RUN) in large and complex failure scenarios (e.g., disaster and cascading failures). However, due to the inherent complexity of multi-layer and multi-domain planning, operation, and optimization in cloud computing, the integration of multiple resiliency techniques adds several research and optimization challenges. Some optimization challenges are how to choose and manage a cost-efficient set of integrated resiliency techniques or architectures while reducing cloud service disruptions and maintaining a tenant's SLO. In Sections V-E and V-F, we categorize some complementary resiliency techniques dealing with some of the aforementioned challenges.

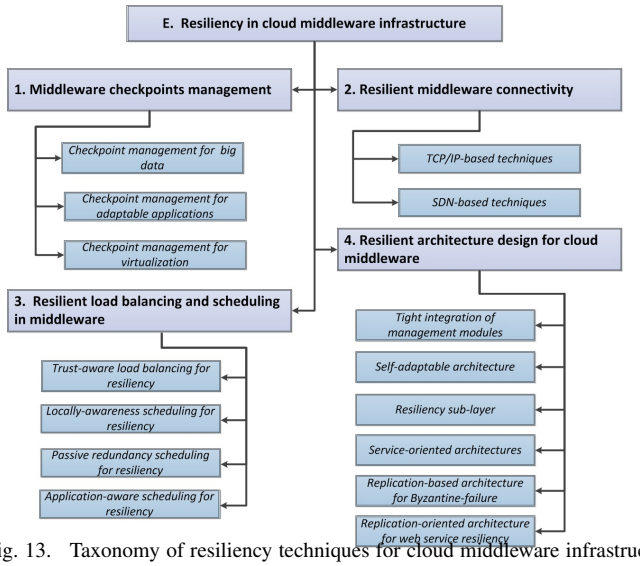

Fig. 13. Taxonomy of resiliency techniques for cloud middleware infrastructure.

E. Resiliency in cloud middleware infrastructure

Middleware is a sublayer implementing key functionalities of the virtualization layer in cloud oriented federated computing and networking systems (CFCNS) infrastructure. Its architecture has been analyzed in [188], where the importance of middleware resiliency for cloud systems was also emphasized. Resiliency approaches for middleware are manifold: checkpoint management, resilient connectivity on top of virtualized components, resilient load balancing and scheduling among virtualized components, and self-adaptable middleware architecture design. Figure 13 introduces the categories for middleware resiliency, and Table IV summarizes the studies on middleware resiliency surveyed in this subsection.

Table IV
APPROACHES FOR RESILIENCY IN CLOUD MIDDLEWARE INFRASTRUCTURE

Techniques and approaches					Contribution for cloud infrastructure resiliency / Type and level of resiliency enforced							
Type	Approach/es	Elements	Strategies	Methods	Main Benefit	Main Challenge	DC	Site	Software*	Physical†	Attacks€	Disaster††
			Middleware checkpoint management									
			Checkpoint management for big data									
RUN	FT-HPC [161]	Cloud	Multiple checkpoint	RE	Low RT for Big data	High overhead	Single	Single	++	++	+	-
RUN	RFTT [162]	Cloud	Mutiple checkpoint/scheduling	RE	Low overhead	Multiple domains	Single	Single	++	+++	++	-
			Checkpoint management for adaptable applications									
RUN	CO [163]	Cloud	Multiple checkpoints synchronization	RE	Low overhead	Multiple domains	Multiple	Multiple	++	+++	+	-
RUN	DAFT [31]	Cloud	Adaptable checkpoint synchronization	RE	High survivability	Multiple domains	Single	Multiple	++	++	++	+
			Checkpoint management in virtualization									
RUN	VFT [164]	CN/VM	Multiple checkpoint and load balancing	RE	Low RT	Multiple domains	Single	Multiple	++	+++	++	-
RUN	BlobCR [165]	Cloud	Multiple checkpoint hardware	RE	Low RT	Multiple domains	Single	Multiple	+++	+++	++	-
			Resilient middleware connectivity									
			TCP/IP-based techniques									
RUN	LLFT [166]	CN/VN	Connection replication	PR/RE	Low overhead	Multiple domains	Multiple	Multiple	++	++	+	+
			SDN-based techniques									
RUN	PicoReplication [167]	CN/VN	Flow checkpointing	PR/RE	Low RT	Multiple domains	Single	Multiple	+	++	+	-
			Resilient load balancing and scheduling									
			Trust-aware load balancing for resiliency									
P/R	TBA [168]	CN/VM	VM traffic migration	PR/FR	High survivability	Forecasting	Multiple	Multiple	+	++	+	+
			Locally-aware scheduling for resiliency									
RUN	BAR [169]	App	Scheduling and allocation	RE	Low RT (App)	Multiple domains	Single	Multiple	++	++	-	-
			Passive redundant scheduling for resiliency									
RUN	FR-BAR [170]	CN/VM	Schedule and allocation	RE	Low RT (VM)	Multiple domains	Multiple	Multiple	++	+++	-	-
			Appplication-aware scheduling for resiliency									
P/R	SHA-APP [171]	VM/App	Schedule and allocation	RE/RM	Redundancy	Multiple domains	Multiple	Multiple	++	++	-	-
PLA	AA [172]	VM/App	Prediction and schedule	PR/FR	SLO awareness	Multiple domains	Multiple	Multiple	++	++	-	-
PLA	HA-APPlace [173]	VM/App	Application replicas scheduling	PR/RM	Multiple tenant	Multiple domains	Multiple	Multiple	++	++	-	-
RUN	RACS [174]	VM/App	Adaptable application scheduling	RE/RM	Low RT	Multiple domains	Multiple	Multiple	++	++	-	-
RUN	CHASE [175]	VM/App	Comprehensive application scheduling	FR/RE	Low RT/Adaptability	Multiple domains	Multiple	Multiple	++	++	-	+
			Resilient architecture design for cloud middleware									
			Tight integration of management modules									
P/R	FTM [176], [177]	Cloud	Integrated resiliency management	All	High survivability	Multiple domains	Multiple	Multiple	++	++	++	-
			Self-adaptable architecture									
DES	Merkaats [178]	Cloud	Redundancy, distribution, and checkpointing	All	High survivability	Multiple domains	Multiple	Multiple	++	++	++	+
DES	COP [179]	Cloud	Redundancy, distribution, and checkpointing	All	High survivability	Multiple domains	Multiple	Multiple	++	++	++	-
			Resiliency sub-layer									
DES	T-Cloud [180]	CN/App	Adaptability, redundancy, and auto-fix	RE/PR/FR	Multiple clouds	High complexity	Multiple	Multiple	+++	+++	++	+
			Service oriented architecture									
PLA	AFTWeb [181]	Cloud	Redundancy, retry, and recovery	All	Adaptability	High complexity	Multiple	Multiple	+++	++	+	-
P/R	HAC [182]	Cloud	Multiple resiliency techniques	All	Multiple layer	Disaster	Multiple	Multiple	+++	+++	+	+
P/R	FTCloud [183]	Cloud	Multiple resiliency techniques	All	Adaptability	Multiple domains	Multiple	Multiple	+++	++	+	+
			Replication oriented architecture for Byzantine failure									
PLA	CFN [184]	Cloud	Failure detection and redundancy	RE/FR	Low RT	High cost	Multiple	Multiple	+++	++	++	-
P/R	Efficient BFT [185]	Cloud	Failure detection and redundancy	RE/FR	Low RT and Cost	Multiple domains	Multiple	Multiple	+++	++	++	-
			Replication oriented architecture for web services resiliency									
RUN	FTWS-O [186]	Cloud	Adaptable service and application redundancy	PR	Low RT	Byzantine attacks	Multiple	Multiple	+++	++	++	-
PLA	DREME [187]	Cloud	Adaptable service and application redundancy	PR/FR	High survivability	Multiple domains	Multiple	Multiple	+++	+++	+	+

Type of approach "DES" design of cloud infrastructure, "RUN" run-time, "PLA" planning of capacity, and "R/P" Run-time or planning or both.
Elements used by the approach, "Cloud" refers to entire cloud or inter-cloud infrastructure, and/or multiple cloud applications, "CN/VN" cloud network with interaction into virtual machines, "App" one or a set of application, "VM/App" interaction VM and cloud application.
Strategies used by each approach following the taxonomy introduced in Fig. 13 and described in this subsection.
Methods for resiliency used, same as in Table III except "All" when the approach uses all methods.
Level of resiliency enforced same as in Table III except "Site" which refers to the entire cloud site.

1. *Middleware checkpoint management:* As seen for VM resilient operation, middleware process resiliency can also be enhanced using checkpointing. The problem to obtain optimal scheduling for checkpoint of multiple components and layers is complex (proven to be NP-hard in [189]), because checkpoint implementation might differ based on the component diversity. This is particularly challenging in large cloud infrastructures due to synchronization, upgrade, and resource management issues. In the following, three resilient checkpoint-management techniques are surveyed:

1.a) *Checkpoint management for big data:* consists in the optimal synchronization of multiple checkpoints to protect and recover big data sets. Some approaches using this technique are the *fault-tolerant high performance cloud strategy (FT-HPC)* [161] and *retrying failure recovery technique (RFTT)* [162]. FT-HPC reduces the interference between checkpoint data and minimizes RT of big data scientific cloud applications. RFTT combines a cloud management system (CMS) with scheduling techniques to recover failed virtual sites. Effective scheduling allows RFTT to work for a large number of services with lower overhead compared to FT-HPC (Table IV).

1.b) *Checkpoint management for adaptable applications:* consists in the use of adaptive synchronization techniques for checkpointing of data for recovery. In short, the synchronization technique is adaptive to the application characteristics and resiliency demands. An approach using this technique is *checkpointing orchestration (CO)* suggested in [163] that minimizes the I/O contention of concurrent checkpoint data from distributed and large cloud infrastructure. Another approach is the *dynamic adaptive fault tolerance (DAFT)* [31] that adaptively coordinates checkpoints and service (i.e., data, connections, and applications) replication to satisfy SLOs.

1.c) *Checkpoint management in virtualization:* combines the checkpointing technique with load balancing and virtualization. *Virtualization and fault tolerance (VFT)* in [164] reduces the recovery overhead in two steps: virtualization and load balancing, and replication of virtualized process, using checkpointing. To reduce the overhead, during the recovery process, VFT uses a cloud manager (CM) and a decision maker (DM) module which detect and block failed PM and software components. Another approach is *BlobCR: efficient checkpoint-restart for HPC applications* in [165] that checkpoints hard disks of each server and reduces the RT compared to previous baseline VM checkpoint approaches.

2. *Resilient middleware connectivity:* extends middleware communication protocols (i.e., TCP/IP) to enhance resiliency of cloud services and applications. Middleware connections are typically set to connect virtual middleware processes using TCP/IP and/or SDN protocols (e.g., OpenFlow) (Table IV).

2.a) *TCP/IP-based techniques:* provide resiliency in cloud middleware infrastructure based on TCP/IP. An approach exploiting TCP point-to-point in combination to UDP multicast is *low-latency fault tolerance (LLFT) middleware* suggested in [166]. LLFT uses passive and active replication of virtual connections synchronized by a leader-determined membership protocol which reduces the complexity of the synchronization. An example of how LLFT works in the case of a system with two clients and two servers is presented in Fig. 14. LLFT is the predecessor of the anycast routing techniques as exploited in the previously discussed [12], [128], [130], [131].

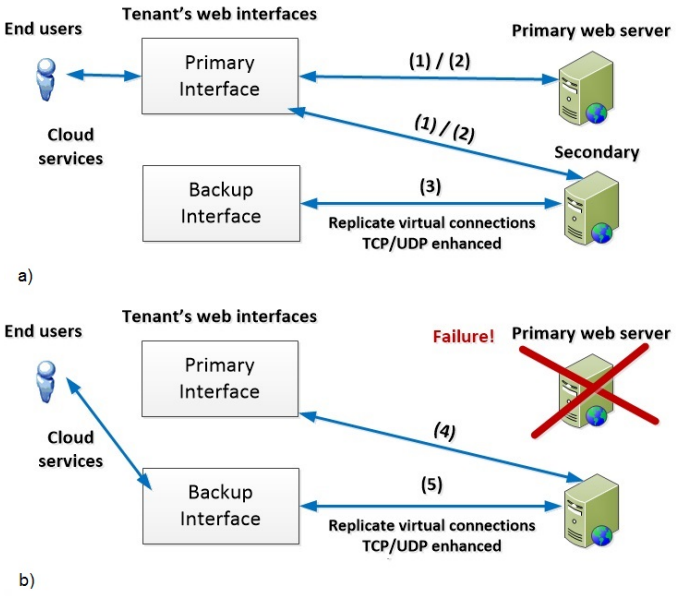

Fig. 14. Example of a low-latency fault tolerance (LLFT) middleware approach. a) *Normal web service operation* where connection (1) sends the regular request, connection (2) sends an acknowledgment to the primary and secondary server, and connection (3) carries a second acknowledgment to confirm the availability of the service. b) Shows an example of *fast service recovery using LLFT* with one web server failure (i.e., physical servers or DC failure), where connection (4) in this case notifies the web server failure and connection (5) acknowledges to keep transferring data.

2.b) *SDN-based techniques:* consist in the extension of SDN protocols to manage/control resiliency of cloud middleware services. SDN is an important enabler for the virtual connectivity required in cloud middleware [121]. For example, *PicoReplication framework for SDN middlebox controller* [167] adds a checkpoint functionality for each independent flow, enabling a novel resiliency scheme on a per-connection basis.

3. *Resilient load balancing and scheduling in middleware:* are techniques that leverage load balancing and scheduling to provide resiliency in the cloud middleware. A brief survey about traffic scheduling and load balancing in middleware is presented in [168], [190] where the main challenges related to virtualization and diversity of cloud applications demands and multi-layer nature of schedule are described. Four main resiliency techniques for load balancing and scheduling in middleware were surveyed based on trust, locality, redundancy, and application awareness (see third section of Table IV). Approaches enabling the resilient load balancing and scheduling maximize survivability of applications and minimize recovery time (RT).

3.a) *Trust-aware load balancing for resiliency:* consists in a load balancing technique using a trust metric to route traffic

between VMs and/or servers. One approach using this technique is *trust-based load balance algorithm (TBA)* suggested in [168] to migrate traffic of overloaded or failed VMs to idle VMs and more trusted data center (i.e., with less failure records) and reduce the chance of total service disruption. This approach complements the trust-aware VM migration discussed before by adding load balancing.

3.b) *Locality-aware scheduling for resiliency:* consists in traffic scheduling using the principle of locality (i.e., closeness) to provide fast service recovery. A cloud data center schedules jobs on servers and VMs with diverse characteristics [168], hence a *balance reduce algorithm (BAR)* for scheduling using data locality is suggested in [169]. BAR schedules and allocates tasks for Hadoop and MapReduce application by minimizing the job completion time gradually while providing low RT in case of server failure (i.e., fast rescheduling jobs).

3.c) *Passive redundancy scheduling for resiliency:* consists in the utilization of passive replications of traffic or jobs on servers and VMs to provide resiliency. An approach using this technique is *failure resilient BAR (FR-BAR)* suggested in [170] to schedule replicated jobs. This approach achieves faster job completion time and RT compared to BAR.

3.d) *Application-aware scheduling for resiliency*: utilizes intelligent scheduling of cloud applications on servers or VMs to provide resiliency. Approaches for application-aware scheduling follow various scheduling principles: availability prediction (i.e., trust), redundant placement, and load balancing. Redundant placement is enabled by load balancing by an approach suggested in [171] that *schedules highly-available applications in a cloud environment (SHA-APP)*. Availability prediction based on stochastic reward nets (SRN) to support the scheduling of application is applied in *Application-level availability analysis (AA)* approach suggested in [175]. A similar approach to SHA-APP, except for the redundancy method and platform, is *high available application placement (HA-APPlace)* suggested in [172]. However, resilient dynamic placement for real-time applications requires more strict time awareness to meet the SLO requirements. Hence, [174] suggests a *reliability assessment for general and real-time cloud computing (RACS)* that includes an algorithm to detect failures and continuously measures the experienced availability of the application components such that the application execution schedule can be adapted to the time-varying requirements (i.e., delay) of the components and services. A more comprehensive approach, combining scheduling methods with application component dependency awareness in static and dynamic scenarios, is *CHASE, component high availability aware scheduler in cloud computing environment* [191]. CHASE extends the HA-APPlace approach and its early version [173] by improving the efficient response to failure scenarios in the cloud infrastructure (i.e., minimizing RT).

4. *Resilient architecture design for cloud middleware.* This category includes a diverse set of studies that aim to provide a comprehensive solution for a resilient middleware architecture design. Some of the studies suggest a tight integration of management modules, or an adaptable resilient architecture, to address the diversity of resiliency demands of cloud services. Other studies suggest a sub-layer architecture to be added between the virtualization and the application layers. Finally, some other studies suggest service and replication oriented cloud middleware frameworks to address Byzantine failures and web services resiliency (Table IV).

4.a) *Tight integration of management modules:* consists in approaches based on the integration of existing management modules from management applications to enforce resiliency. One approach using this technique is *failure tolerance manager (FTM)* proposed in [176], [177]. FTM integrates three management modules to enable comprehensive and adaptable resiliency in large cloud infrastructure: (i) *Resource manager* to allocate resources avoiding congestion and failures; (ii) *Replication manager and messaging monitor* to enforce the active replication and grouping of contents; and (iii) *Fault masking and recovery manager* to recover failures while masking them to end users. This integration of management components enforces failure forecasting, removal, protection, and recovery of the infrastructure (see Table IV).

4.b) *Self-adaptable architectures:* are those designed to provide adaptability of the cloud middleware based on security and protection requirements. *Merkaats* [178] creates idle and changing target applications to confuse the attacker and to keep the redundancy in case of server failures. The Merkaats architecture integrates anomaly detection, data replication, and checkpointing to provide protection and restoration. A similar architecture is the *control operations plane (COP)* suggested in [179] that uses a security scheme similar to Merkaats but with additional restoration schemes including physical components (e.g., retry and reboot techniques).

4.c) *Resiliency sub-layer:* consists in adding a specialized middleware sub-layer to enhance resiliency in cloud infrastructures. For example, the *trusted cloud (TCloud)* sub-layer proposed in [180] aims to detect, estimate, and mitigate failures and Byzantine attacks in federated clouds, by emulating the functionalities of a virtual switch that enforces traffic redundancy and adaptability between cloud application and VMs and self-healing functionalities (e.g., auto-corrections). The virtual switch can also be replicated to enhance the level of resiliency for distributed applications. Figure 15 presents the virtual switch architecture of TCloud with an example failure in one cloud provider. TCloud reallocates resources and reassigns trust in components (following the red arrows).

4.d) *Service-oriented architectures:* are middleware architectures designed to provide differentiated resiliency based on the QoS and/or SLO requirements of the supported services. *Adaptive QoS-aware fault tolerance strategy for web services (AFTWeb)* [181], for example, selects and executes the best recovery technique based on the SLO of the application to be recovered. Some of the specific strategies implemented in AFTWeb are: retry and reboot, recovery blocks, and active replication. A similar approach is *high availability configuration for cloud (HAC)* in [182]. HAC selects and adapts the same types of fault-tolerance strategies used by AFTWeb

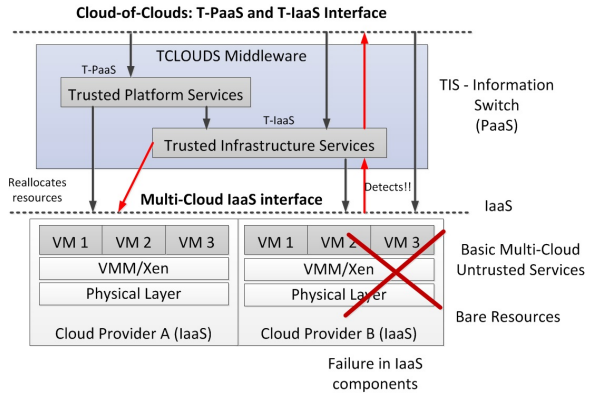

Fig. 15. Representation of the high-level logical functioning of TCloud middleware sub-layer [180]. Following the red arrows, first (during failure-free system operation) TCloud duplicates the image of VM 2 of cloud provider B into the VM 2 of the cloud provider A. Second, following the black arrows, when a failure in provider B occurs, TCloud activates VM 2 of provider A and coordinates the migration of the service towards the safe copy in provider A.

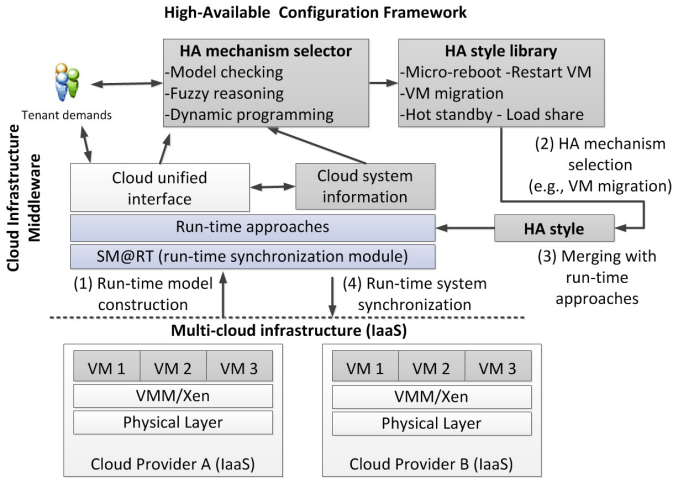

Fig. 16. High-Available Cloud (HAC) configuration framework. Steps followed by HAC: (1) the SM@RT module requests the initiation of the run-time construction model based on current system status; (2) after getting the information from the cloud unified interface and cloud system, the HA mechanism selector chooses the best HA style (i.e., the resiliency strategy) for the requesting tenant application; (3) then the HA style passes (merges) this information to the run-time approaches; and (4) finally SM@RT synchronizes the infrastructure status based on new run-time information.

to control VM reload, replication, and relocation. Figure 16 describes the HAC framework, including an example of the configuration process: as we can see from the figure, HAC has a similar structure to TCloud (Fig. 15) but it also uses a module for run-time synchronization called SM@RT. Another approach similar to HAC is *failure tolerance cloud (FTCloud)* [183] which uses a ranking mechanism to support a resilient resource allocation for tenants with an adaptable selection of the resiliency strategies based on the SLO of each cloud application and/or services.

FTM, TCloud, AFTWeb, and HAC provide similar functionalities, however the main differences are in the selection and implementation of the fault-tolerance strategies. For example, HAC uses an effective combination of multiple fault-tolerance strategies supported at different cloud layers (i.e., virtualization and application layers).

4.e) *Replication-based architecture for Byzantine-failure resiliency:* consists in architecture design for cloud service replication to avoid failures cased by Byzantine attacks[2]. An architecture called *cloud computing fault net (CFN)* is suggested [184] to detect Byzantine failures in large cloud computing infrastructure using petri-nets and Byzantine failure patterns at the cloud middleware infrastructure. CFN shows that, by replicating data and content into $3f + 1$ server replicas, CFN can tolerate up to f failures. However, replication increases cost in resources and management, hence [185] proposed an efficient approach to reduce the number of replicas required for Byzantine-attack resiliency to $2f + 1$ servers.

4.f) *Replication-based architecture for web service resiliency:* consists in a replication mechanism to enforce web services resiliency in cloud middleware. A passive-replication approach using business intelligence is *fault tolerance web service orchestration (FTWS-O)* in [186]. The business intelligence is used to identify critical web services which need to be replicated. This approach reduces the number of replicas, but it might not be able to protect against advanced Byzantine attacks. Another replication-based architecture is the *diversified replica execution and monitoring environment (DREME)* [187] that adds monitoring and lightweight virtualization (i.e., operating system virtualization) to replication in order to enable web service execution in different locations, allowing to overcome server failures.

Discussion of benefits and challenges for resiliency techniques in cloud middleware infrastructure: Table IV describes and compares approaches on resiliency in cloud middleware infrastructures and discusses some benefits and challenges. The main benefits for cloud infrastructure designers and providers are: (i) multi-layer integration and management of resiliency techniques (i.e., resilient architecture design for cloud middleware) by supporting an efficient integration and management of multiple resiliency techniques; (ii) comprehensive resiliency planning and operation (i.e., resilient checkpointing management and resilient load balance and scheduling). The main challenges for cloud infrastructure designer and providers are: (i) high computational complexity required by the resource optimization and (ii) measurement and estimation tools required. To deal with both challenges, measurement and estimation techniques are essential (Section V-F).

F. Measurement and estimation of resiliency in cloud computing

As in any system, measuring and estimating resiliency is an important aspect for cloud computing infrastructure and applications. Several works focusing on measurement and estimation were surveyed in [81]. In this section, we introduce

[2]Byzantine failure resiliency approaches focus mostly in server, cloud infrastructures, and applications; however, some approaches suggest Byzantine-failure tolerance also in cloud middleware infrastructure.

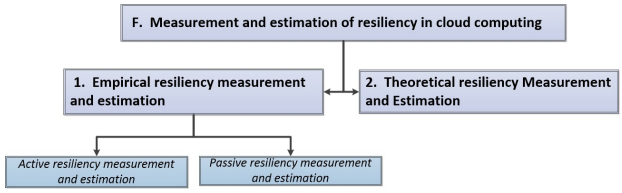

Fig. 17. Taxonomy of measurement techniques for cloud computing resiliency.

some important works by categorizing them in two groups: empirical and theoretical approaches.

1. *Empirical resiliency measurement and estimation:* are used for real cloud computing infrastructures which can be *active* or *passive*. Active approaches induce failures to measure their effect, while passive measurements amount to logging what failures actually happen as is.

1.a) *Active resiliency measurement and estimation:* The injection of failures and anomalies in a cloud infrastructure can measure the response to the injected failures. The common goal is to forecast the recovery time (RT) and survivability capability and remove weaknesses in the infrastructure (removal) and choose the correct resiliency approach.

The failure injection can be programmed for each tenant or for the entire cloud infrastructure to measure a large and complex failure recovery. Some approaches focusing on this strategy are: *Failure testing service (FATE) and declarative testing specifications (DESTINI)* [192], and *PreFail, a programmable failure-injection tool* [193]. PreFail addresses the complexity of multiple-failure recovery testing, by using policies to prune the large space of all possible failure combinations to, e.g., a realistic, representative subset for testing purposes. FATE systematically tests recovery under multiple failures, while DESTINI specifies precisely the expected recovery behavior. However, the complexity of cloud infrastructure and applications is the main limitation of both techniques.

To address large testing of resiliency, *failure-as-a-service (FaaS)* defines a set of tools that enable large-scale failure drills in real deployments [194]. However, since FaaS' injection and measurement are limited to one cloud infrastructure provider, it has been extended to *failure scenarios as a service (FSaaS)* [195], large-scale measurements of multiple cloud infrastructures or federated clouds.

Examples of empirical measurement and estimation in virtualized environments are *cloud dependability analysis (CDA)* [196] and *D-Cloud* suggested in [197]. CDA analyzes the correlation of various performance metrics with failure events in virtualized and non-virtualized systems.
D-Cloud enables automated resiliency test procedures, using VMs with a fault injection mechanism (FaultVM): this enables tests including hardware faults by emulating hardware faults by FaultVM.

1.b) *Passive resiliency measurement and estimation:* The inclusion of passive applications or devices can capture abnormal behavior or performance in regular run-time of cloud infrastructures and applications.

Empirical passive measurement of failures (in terms of frequency, reparation time, and cost) has been performed in operational cloud infrastructure in studies by, e.g., Microsoft [28] and Google [18]. Similarly, resiliency was measured in the high performance computing site of Los Alamos National Laboratory over a period of 9 years (recording various failure types and their root causes) [198].

2. *Theoretical resiliency measurement and estimation:* these methods start from system models and then assess resiliency metrics by numerical methods or simulation programs. Model parameters are set in accordance to reality based on previous experiences, known patterns of events, etc.

Ghosh *et al.* [199] present a theoretical measurement analysis of resiliency in cloud computing infrastructure in terms of job rejection rate and capacity provisioning delay. The main focus of the proposed method is the use of a stochastic reward network (SRN) based on petri nets which is used in some approaches for cloud infrastructure and application resiliency for estimating the resource availability to place VMs or contents. SRN is used for resilient cloud application deployments in the cloud middleware infrastructure [172].

Resilience for *virtual infrastructure management systems* (e.g., VM management, Xen hypervisor) is studied in [200] by suggesting evaluation metrics and selecting the most resilient and scalable logical topology (i.e., controller and VM). Between a centralized, hierarchical, and peer-to-peer logical topology, the authors conclude that a hierarchical topology is the most resilient, while a peer-to-peer topology is the most scalable. Adding the *resource management system* (RMS) into the logical topology for physical resource allocation, peer-to-peer becomes the most resilient and scalable topology for cloud virtualization [201].

The failure estimation in cloud infrastructure components typically is based on crude metrics, such as MTBF or just specifying the mean number of failures. Hence, some studies suggest more accurate probabilistic models based on the "*risk assessment*" of elements in the cloud infrastructure by combining two key parts: (i) the probability of occurrence of a failure or an attack and (ii) the expected damage in case of the occurrence [202]. For the case of failures in cloud data centers and long-distance optical networks infrastructures, empirical studies and hazard maps of potential natural disasters might provide estimation of the probability of occurrence and expected damage [97]. Attacks can be predicted using probabilistic models to assess the risk [203]. Studies in [97], [156], [204], [205] analyze the risk assessment measurement in communication networks, then [206] suggests the integration of the risk assessment in the entire cloud computing infrastructure.

VI. Resiliency in Cloud Computing Applications

A cloud computing application typically has a web interface that provides access to a set of core objects and data objects placed with one or many cloud infrastructure providers. *Core objects* form the logic of the application: programmed functions or blocks of instructions, or classes executed by

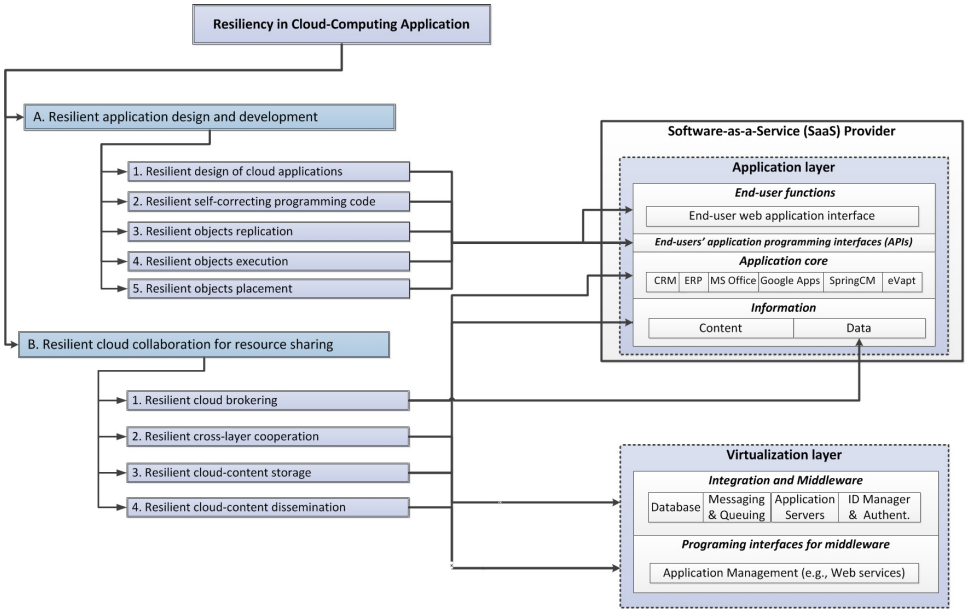

Fig. 18. Taxonomy of surveyed approaches for resiliency in cloud computing applications.

an application. These core objects can be developed using multipurpose programming languages (e.g., Java, C++, C, Python) and/or with some specific cloud or distributed system languages/frameworks (e.g., MapReduce [207], DryadLINQ [208], Pregel [209], or X10 [210]). *Data objects* are the content used and/or produced by such logic: related raw data, database records, content files, etc., stored and managed by an application. Each cloud application can be placed and executed in multiple cloud infrastructures that may be geographically distributed and/or could be multi-tenant.

To provide application resiliency against possible infrastructure failures (as well as cyber attacks on the level of the application layer), both the *core* and *data* objects need to be protected. Conceptually, both object types can be protected against failures adopting similar ideas. We categorize the surveyed studies according to their core ideas into (See summary in Table V): (1) resilient application design and development and (2) resilient cloud collaboration for resource sharing data/content. The next subsections VI-A and VI-B discuss each category. Figure 18 summarizes the overall classification, and also shows how the first category (1) resides entirely in the application layer, while (2) also involves the virtualization layer.

A. Resilient application design and development

At the application layer, we identify five main resiliency approaches: application design, object self-correction, object replication, object execution, and object placement.

1. *Resilient design of cloud applications:* Cloud application resiliency can be enforced in the early stage of design, where best practices can be adopted to ensure resilient operation. For instance, the *provision, assurance and auditing (PAA) framework* [211] suggests the addition of control functionalities in the design of cloud applications to support the auditing of security and data integrity. The auditing functionality will allow the application manager and tenant to evaluate, investigate, and mitigate future failures and attacks.

Another example is the categorization of best practices in a so-called *Byzantine fault-tolerance (BFT)* catalog including previous Byzantine failures in web services to assist application designers [212]. The BFT catalog includes patterns of critical interactions between sophisticated web service applications with lessons learned for the application designer to avoid well-known and preventable errors, attacks, and failures.

2. *Resilient self-correcting programming code:* Applications developed and implemented using this methodology enable self-correction in critical code parts of an object to ensure resiliency (i.e., self-healing).

In *resilient X10* (X10RT), self-healing is realized by maintaining code snapshots, and falling back to "the best previous working code" [213]. X10RT extends the cloud computing application programming language X10 by adding a run-time, i.e., "memory resident algorithm" [213] that detects and repairs the logic of any program (i.e., the *core* object) written in X10 or in a multi-purpose language. Tests in cloud applications using X10RT showed good recovery capabilities for single virtual node (VN) failures with a small additional overhead.

3. *Resilient object replication:* This method enforces replication of critical objects (both *core* and *data*) to protect the application and user's data/content. Object replication can be passive (i.e., a scheme with one standby backup object per primary, 1:1 protection) or active (i.e., N-version programming, 1:N protection). In both cases, replication also requires security and synchronization to provide consistent application protection and recovery.

A Java-based active approach is *immune system (ImSys)* [214], which uses CORBA for active object replication to mitigate and recover malicious attacks and failures. ImSys relies on a secure multicast protocol to manage secure syn-

Table V
APPROACHES FOR RESILIENCY IN CLOUD COMPUTING APPLICATION

Techniques and approaches					Contribution for cloud application resiliency / Type and level of resiliency enforced							
Type	Approach/es	Elements	Strategies	Methods	Main Benefit	Main Challenge	Site	App	Software*	Physical†	Attack$^{\epsilon}$	Disaster††
			Resilient application design and development									
DES	PAA [211]	App	Application design guide	FR/RM	Planning	Depend on design	Single	Single	+	+	-	-
DES	BFTCat [212]	App	Application design guide	FR/RM	Byzantine attack resiliency	Depend on design	Single	Single	+	+	+	-
P/R	X10RT [213]	App	Resilient self-correction code	RE	High survivability.	High overhead	Single	Single	++	++	-	-
P/R	ImSys [214]	App	Resilient object replication	PR/RE	High survivability	Multiple tenants	Single	Single	++	++	-	-
P/R	MillWheel [215]	App	Resilient object replication	PR/RE	Low RT	Multiple tenants	Single	Single	++	+++	+	-
P/R	SCO [216]	App	Resilient object execution	PR	High survivability and adaptability	Multiple domains	Multiple	Single	+++	+++	+	-
P/R	RDD [217]	App	Resilient object execution	PR	High survivability and adaptability	Multiple domains	Single	Single	+++	++	+	-
RUN	FTCloud [183]	App	Resilient object execution	FR/RM	High adaptability	Complexity	Multiple	Multiple	+++	+++	++	+
P/R	ADAPT [218]	App	Resilient object placement	FR/RM	High adaptability	Complexity	Single	Multiple	++	++	+	-
			Resilient cloud collaboration for resource sharing									
RUN	iRec [219]	Site	Resilient cloud brokering	FR/RM	Inter-cloud resiliency	Complexity	Multiple	Multiple	+++	++	++	+
RUN	1:1 SP [220]	Site	Resilient cloud brokering	FR/RM	Inter-cloud resiliency	Complexity	Multiple	Multiple	+++	++	++	+
RUN	SACC [22]	Site	Resilient cloud brokering	FR/RM	Inter-cloud resiliency	Complexity	Multiple	Multiple	+++	++	++	+
RUN	CFTM [47]	Site	Cross-layer cooperation	PR/RE	High adaptability	Multiple domains	Multiple	Multiple	+++	+++	++	+
RUN	SACC [22]	Site	Resilient cloud storage	PR/RE	Low RT and high survivability	High overhead and cost	Single	Multiple	+++	+	++	+
RUN	RACS [221]	Site	Resilient cloud storage	PR/RE	Low cost and high survivability	High overhead	Single	Multiple	+++	+	++	+
RUN	NCCloud [222]	Site	Resilient cloud storage	PR/RE	Low cost and RT	Complexity	Multiple	Multiple	+++	++	++	+
RUN	SKUTE [223]	Site	Resilient cloud content dissemination	PR/RE	High adaptability	Complexity	Multiple	Multiple	+++	++	+	+
RUN	DepSky [23]	Site	Resilient cloud content dissemination	PR/RE	Inter-cloud resiliency	High cost	Multiple	Multiple	+++	+++	++	+
RUN	SPANStore [224]	Site	Resilient cloud content dissemination	PR/RE	Low cost and RT	Complexity	Multiple	Multiple	+++	++	++	+
RUN	BlueSky[225]	Site	Resilient access cloud application	PR/RE	Low RT	Complexity	Multiple	Multiple	+++	++	++	+
RUN	μLibCloud [226]	Site	Resilient access cloud application	PR/RE	Low RT	Complexity	Multiple	Multiple	+++	++	++	+

Type of approach "DES" design of cloud infrastructure, "RUN" run-time, "PLA" planning of capacity, and "R/P" Run-time or planning or both.
Elements used by the approach, "Cloud" refers entire cloud or inter-cloud infrastructure, and/or multiple cloud applications, "CN/VN" cloud network with interaction into virtual machines, "App" one or a set of application, "Site" set of cloud applications of one cloud application provider.
Strategies used by each approach following the taxonomy introduced in Fig. 18 and described in this subsection.
Methods for resiliency used, same as in Table III.
Level of resiliency enforced *Survivability in different levels of cloud computing application failures, "+" minor failure in process and data, "++" failure of multiple process and data (i.e., cloud application or service unreachable temporary), "+++" failure in the whole application and site (i.e., cloud application or service corruption with loses).
†Survivability in different level of cloud infrastructure failures, "+" failure of single component of the DC including one or more servers and communication devices in the intra/inter DCN, "++" failure of one DC and/or links of the inter DCN, "+++" failure of many links and components of the inter-DCN, and more than one DCs are disconnected or an entire cloud infrastructure.
$^{\epsilon}$Survivability in different level of attacks, "+" attacks on one tenant and minor services disruption "++" attacks affecting a DC with some data loses in more than one tenant, difficult to mitigate, "+++" sophisticate attack with important data losses in several tenants, the attacker might control some important equipment.
††Survivability in case of disaster and post-disaster cascading failures, "+" one disaster with low risk of cascading failure, "++" more severe disaster with moderate risk of cascading failures, "+++" large disaster with high risk of cascading failure.

chronization of messages and data between object replicas.

Another active data replication using the checkpoint approach for deadline driven (i.e., time-constrained) applications is *MillWheel* [215]. Tenants or end users using MillWheel write application objects as individual nodes in a directed graph with arbitrary topology (where an edge from node A to B indicates that B processes A's output). Data objects are replicated and checkpointed in continuous flows between edges of the graph (similar to 1+1 protection) in such a way that, even if any node is corrupted or lost, the application will not be affected. The MillWheel approach is very useful for many Google cloud applications (e.g., Google Ads, Google Street View, etc.). The overhead of this method (i.e., additional resources for the object replicas and their synchronization) may limit its feasibility for large cloud applications.

4. *Resilient objects execution:* This method provides protection and recovery for distributed cloud applications without the requirement of full duplication of objects for resiliency purposes. It adopts a checkpointing approach: in which so-called "movable data containers" are created to add checkpoint, adaptability, and security functionalities for recovery and protection.

A sample approach that encapsulates critical data objects is *self-controlled objects (SCO)* [216]. SCO containers are generated by end users through wizards to be distributed across different application domains inside and outside a cloud provider. Experimental results show how tested distributed applications with SCO may survive (physical and/or VM) server failures [216].

Resilient distributed data sets (RDDs) is an object programming abstraction enforcing "read-only, partitioned collection of records or data sets" to store critical information and to create self-protected applications [217]. An RDD programming method enforces object protection by storing a data set in a read-only part of the memory or in different storage locations of the cloud. RDD elements can be divided and distributed over multiple VMs by using keys as means to identify them. RDD can be used in any general-purpose language and distributed-system-based language.

The main difference between SCO and RDD is that SCO stores an entire critical object in movable containers while RDD creates "metadata" of critical objects in a read-only and fixed location in memory or in safe storage. This metadata can be used as a checkpoint to recover the application or each individual object. The main weakness of the suggested approaches is their scalability and dependency on the application designer and developer's skills.

5. *Resilient object placement:* These methods use failure prediction or detection before storing or relocating critical objects, so as to pick locations with minimal expected failures. An approach using such ranking to enable resilient object placement is the "component ranking framework for fault-tolerant cloud application" (FTCloud) [183]. FTCloud builds a ranking between cloud components to identify and test each cloud component and identify the most available. The higher-ranked components are used to place the objects of the cloud application. In case of failures or congestion, FTCloud provides an algorithm to recalculate the rankings and suggest a new object placement. This approach can work with any application development framework.

A similar approach to enforce resilient objects placement is "availability-aware MapReduce data placement mechanism" (ADAPT) [218]. ADAPT helps MapReduce- and Hadoop-based applications to mitigate the impact of sparse resource allocation and node failures (e.g., server or VM failure) by dynamically placing objects into working VMs and servers to increase redundancy. To relocate objects in case of resource exhaustion or failure, ADAPT detects any changes in the availability of cloud components and suggests relocations, if necessary. Experiments show a 30% improvement of the availability when ADAPT is used [218].

The approaches in this subsection have some significant limitations to handle multiple applications and severe failures (i.e., multiple cascading failures as in a disaster): survivability of "application design and development approaches" is mostly limited to moderate failures and attacks on cloud applications which include single-server failure and intrusion.

B. Resilient cloud collaboration for resource sharing

The collaboration approaches we now discuss have cloud providers and tenants sharing computational resources for application and data/content replication, distribution, and placement. Approaches of this group extend the cloud application survivability to more severe failures and attacks which include multiple server or data center failures or an entire cloud provider infrastructure disruption or under cyber attacks (Table V).

Collaboration along tenant's applications hosted by the same cloud provider or different cloud providers can enable resource sharing and data/content protection and recovery against severe failures scenarios [227]. This scenario also requires data/content to be replicated, placed, and synchronized.[3] Most surveyed approaches are located in the application layer by interacting with the upper sub-layer of the virtualization layer (i.e., the sub-layer of integration and middleware).

We identified five categories of studies, as discussed below.

1. *Resilient cloud brokering:* Cloud application brokering is an approach to automatically provide and manage resources for an application, including its allocation and load balancing [227]. Surveyed studies focus on two aspects: providing (i) *recommendation* to relocate an application in a more resilient infrastructure (i.e., forecasting and removal), and (ii) a *complete brokering* to adapt the resource exhaustion and failure (i.e., restoration).

An approach for *recommendation* to support safe multiple cloud resource replication to enforce resiliency is the "failure independent cloud application placement recommendation system (iRec)" suggested in [219]. iRec acts as an intermediate

[3]Note that data/content can be a service, a set of services, an application, a set of applications, a database, or a set of databases managed by a tenant or by multiple tenants.

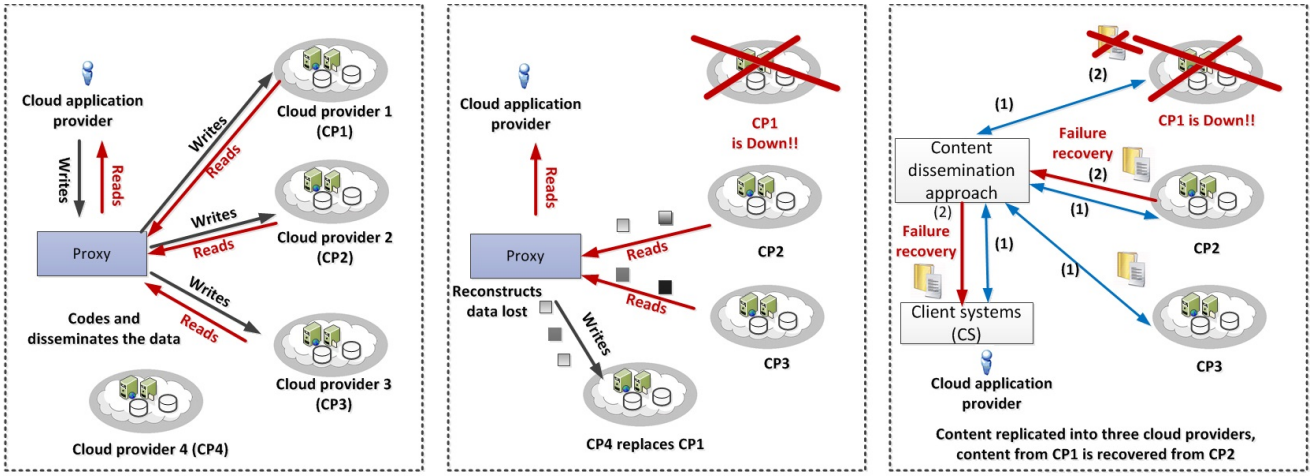

Fig. 19. Resilient content/data coding. a) Normal operation of proxy which codes data between cloud provider (writing and reading). b) Example of failure scenario where data affected by a failure in cloud provider 1 can be recovered by the proxy. c) Content replication approach, where (1) the content is replicated and disseminated in three cloud providers and (2) where a failure in cloud provider 1 can be easily be recovered using cloud provider 2.

entity between applications and cloud providers to suggest the best application placement. The redundancy (and thus also cost) of application replica is reduced substantially because iRec manages the information about status of each cloud provider, including the frequency of failures, congestion, SLO, and available resources to share. Another important functionality of iRec is the trust and security enforcement by hiding information around tenants and cloud providers.

Another approach suggesting a *complete brokering* for application placement by exploiting the collaboration between private and public clouds is 1:1 service protection (1:1 SP) proposed in [220]. 1:1 SP provides protection of private clouds (i.e., entire cloud application providers) by using public cloud resources. The approach performs resource brokering using Markov chains to forecast public cloud availability. Once it defines the most available public cloud, 1:1 SP places a replica of each application to enforce resiliency.

A more complete brokering scheme is "self-adaptive cloud collaboration" (SACC) [22]. SACC first evaluates the "health" of cloud components (e.g., SaaS application components and PaaS/IaaS infrastructure components) by measuring the congestion and history of failures. While receiving and processing cloud application requests, SACC selects the best collaboration links (e.g., virtual connections and agreements) between cloud providers before placing the application. Finally, SACC uses the application placement request, the health of each component, and best collaboration links to decide where to place the application.

After placing the applications, SACC observes the changes in health of each component (e.g., failure or corruption or resource exhaustion) and changes in the application requirement (e.g., increase of resource usage and SLA requirements) to relocate applications and contents.

2. *Resilient cross-layer cooperation:* An approach for multi-layer cooperation for resiliency to detect and recover failures is "collaborative fault tolerance management" (CFTM) [47]. CFTM considers three levels of failures: application, VM, and physical machine. The failures are detected by sensors in the applications (i.e., at the SaaS tenant side) and VMs (i.e., at the provider side). The most severe situation is a physical machine (i.e., server) failure: the provider will recover the VMs of the failed server in another similar server and ask the tenant to re-install the application. If the failure occurs in the application or tenant domain, the provider will offer mechanisms to migrate or replicate the application.

3. *Resilient cloud content storage:* Application and content placement between cloud providers requires resilient data storage methods. Some surveyed approaches enable network/data coding techniques to provide data/content recovery. The data/content coding for recovery entails division and distribution of data/content in coded blocks enhanced with "erasure segments" that allow the reconstruction of the entire content from only part of the blocks [228]. Resiliency in cloud computing has two levels of coding: application/middleware coding for inter-cloud storage, and system coding for data recovery in hardware components.

For example, RAID coding techniques in disks and files for resiliency are very popular in the hardware industry. RAID-based coding with a proxy for application and content resiliency is exploited by "redundant array of cloud storage" (RACS) [221]. RACS splits and distributes user data in blocks (i.e., buckets) with minimal replication requirements. The proxy manages block distributions, the reading and writing operation between cloud providers (e.g., Amazon S3), as sketched in Fig. 19). The cost reduction is evident due to the reduced amount of redundant information required compared to RAID-5.0 type of replication. However, the processing time in normal operations, and also the recovery time (RT) when failures occur, can be high.

To reduce the processing time, a more efficient regenerat-

ing code is suggested in "network-coding cloud" (NCCloud) [222]. The main strength of NCCloud is the correction coding, called functional minimum-storage regenerating code (F-MSR), which allows to maintain the data redundancy with less traffic and overhead, reducing the recovery time (RT). NCCloud provides the same resiliency and response time obtained by the RAID 6.0 in multiple clouds. NCCloud is designed to recover more severe failure and data corruption compared to RACS by using more complex regeneration code. Experimental results with Amazon S3, RackSpace, and Windows Azure cloud providers show the advantages of NCCloud in term of recovery time and cost [222].

Figure 19(a) presents an example of NCCloud's read and write steps in normal operation, and Fig. 19(b) shows a case of one cloud provider failure.

4. *Resilient cloud content dissemination:* Cloud oriented content delivery network (CCDN) is the cloud implementation of content delivery networks (CDNs), which are specialized networks connecting distributed storage servers for content access and replication [229]. CCDN is a fundamental concept shared by all the following proposals: Fig. 19(c) shows how, in order to enhance resiliency, tenants (i.e., SaaS application providers) can exploit content dissemination by replicating data/content placement in the infrastructure of one or many cloud providers (e.g., Amazon S3).

A self-adaptive replication approach, called "scattered key-value store" (SKUTE), was suggested in [223]. SKUTE provides cost-efficient geographically-distributed data/content replication. The approach adapts the replication scheme based on application requirement and types of failures (e.g., physical, software, and disaster).

A more comprehensive proposal is the "dependable and secure storage in a cloud-of-clouds" (DeepSky) [23]. Besides content distribution and replication, DeepSky performs coding of critical data (i.e., resilient data coding), secure data interchange, and Byzantine quorum (i.e., the best BFT method [230]) content distribution. Figure 19(c) shows the main functionalities of DeepSky. A test of DeepSky with four commercial cloud storage providers (Amazon S3, Microsoft Azure, Nirvamix, and RackSpace) showed promising results to reduce the service disruption. An improved version of DeepSky is presented in [231].

Another cost-effective approach enabling geo-replication of data/content through multiple cloud providers is "storage provider aggregating networked store" (SPANStore), proposed in [224]. SPANStore integrates geo-distributed federated clouds by replicating data/content using an asymmetric Quorum. The main goal of this approach is to minimize the application cost while minimizing the latency and maximizing the consistency and resiliency. The application content replication and distribution performed by SPANStore enforce more time- and cost-efficient resiliency compared SKUTE. Experimental results of SPANStore with Amazon S3, Microsoft Azure, and GCS cloud providers showed a cost-efficient protection for a case of a server and an entire cloud provider failure [224].

Although the three approaches are quite similar, DeepSky enables security, protection for case of Byzantine attacks, and network coding, that are not enabled by SPANStore and SKUTE. However, DeepSky can face scalability issues in large and highly-diverse application demands and cloud providers.

5. *Resilient access to cloud applications and content:* Once content has been safely placed, resilient remote user access to it is critical. Several approaches enabling resilient content/data access focus on this aspect. BlueSky is a file-system-based approach using a "proxy architecture" to guarantee resilient access/connectivity to data at application layer [225]. BlueSky uses the proxy-based architecture (similar to NCCloud) as another layer in the hierarchy to perform checkpoint backups. The checkpoint stores the snapshot of the root in the file system to support a protection and recovery strategy when a failure occurs.

An approach enabling dissemination and access to data/content for resiliency using Apache LibCloud is "high availability and uniform access to multiple cloud storage" (μLibCloud) [226]. μLibCloud works at the tenant or end user side, collecting and distributing data from and to different cloud providers using a simplified coding technique similar to DeepSky. Experiments over seven cloud providers showed that μLibCloud significantly increases availability in case of a catastrophic failure (i.e., disaster or cyber attack), with low cost and without overhead. Checkpoints increase the robustness of BlueSky, and a network-coding technique optimizes data access and distribution in μLibCloud.

VII. Analysis and Conclusion

In this section, we conclude our survey with an overall summary of cloud resiliency approaches, and a short analysis of major trends and open problems in cloud computing resiliency.

A. Summary of cloud resiliency approaches

The first section of this survey introduced the main characteristics of cloud computing, the service model, the architecture, the challenges of resiliency, and some preliminary definitions used throughout this work. In the next two sections, we categorized and introduced several approaches to resiliently design and operation of cloud infrastructures and applications. We also included measurement and estimation techniques for cloud computing resiliency. Here, we provide an overall summary of both sections, with a global overview of all techniques (spanning servers, networks, middleware, and the application layer) in Table VI, which compares them in terms of methods, goals, and survivability. We coarsely refer in the following to basic, mid, and high resiliency level to indicate both the quality of the resiliency (recovery time and availability) and the coordination capabilities among the computing and networking domains of the cloud system. From Table VI, we observe the following:

1. *Server and network resiliency* provides the fundamental survivability for the cloud components (servers and network equipment). The degree/level of resiliency varies among the different approaches. A *basic* resiliency level is achieved by

Table VI
SUMMARY OF TECHNIQUES FOR CLOUD COMPUTING RESILIENCY

Type and usage			Methods and goals				Survivability enforced			
Main categorization of resiliency techniques	Domain	Expected usage	Main methods	Min. RT	Cost	Resiliency strategies	Software failure∗	Byzantine attack†	Physical failure$^{\epsilon}$	Disaster failure††
Resiliency Techniques for Cloud Infrastructure										
Resilient facility design and operation	Infrastructure designer and operator	Robust facility design and operation	Redundant power and secure access, and operation	X		PR/RM FR/RE	[+]	[+]	++	[+]
Resiliency in servers										
Resiliency in physical servers	PM designer and operator	OS, PM, NAS, SAN design and run-time	Redundancy, fencing, reset/reboot, EDCC, and RAID	X	X	RM/PR RE	+	+	+	
Resiliency in virtual server	Cloud provider data centers	Inter-PMs and OSs run-time	VM migration, replication, placement, and checkpoint, VS coding and design	X	X	PR RE/FR	++	[+]	++	[+]
Resiliency in networks										
Resiliency in data center networks	Cloud designers and providers	Intra-DCN and inter-DCN design and run-time	Replication of server ports, switches, and routes		X	PR/RE FR/RM	[+]		+++	[+]
Resiliency in virtualized networks	Cloud designers and providers	Intra-DCN and inter-DCN design and run-time	Topology cut-aware and augmentation, service and multi-layer resiliency	X		RM/PR RE	[++]	[+]	+++	+
Resiliency in cloud integrated infrastructure										
Resiliency in cloud oriented FCNS infrastructure	Cloud provider with multiple data centers	Data center and content planning and run-time	CD, VA, Place, AnyCast, DR, and failure detection	X	X	RE/PR RM/FR	+	+	+++	+
Resiliency in cloud network	Cloud provider with multiple data centers	Data center and content planning and run-time	VM/VN survivability, VBN, AnyCast, backup sharing, DR, and detection	X	X	PR/RE RM/FR	+++	+	+++	++
Resiliency in cloud middleware infrastructure										
Middleware checkpoint management	Cloud provider with multiple data centers	Middleware infrastructure run-time	Multipled checkpoint orchestration	X		RE	++	+	++	[+]
Resilient middleware connectivity	Cloud provider with multiple data centers	Middleware infrastructure run-time	Redundant connection and flow checkpointing	X		RE/PR	++	+	++	[+]
Resilient load balancing and scheduling in middleware	Cloud provider with multiple data centers	Middleware infrastructure planning and run-time	Trust, redundancy, allocation, scheduling, and load balancing	X		RE/FR PR/RM	[++]		[++]	
Resilient architecture design for cloud middleware	Cloud provider with multiple data centers	Middleware infrastructure design, planning, and run-time	Management integration, self-adaptable, service-oriented replication architecture	X		RE/FR PR/RM	+++	[++]	[+++]	[++]
Measurement and estimation of resiliency in cloud										
Empirical resiliency measurement and estimation	Cloud provider and infrastructure	Cloud computing planning and run-time	Failure injection and failure-as-a-service (FaaS)	X		FR/RM	[++]	[++]	[++]	[+]
Theoretical resiliency measurement and estimation	Cloud provider and infrastructure	Cloud computing planning and run-time	Failure simulation and modeling	X		FR/RM	[++]	[++]	[++]	[+]
Resiliency in Cloud Computing Application										
Resilient application design and development	Cloud site and applications	Design of cloud applications and/or cloud sites	Objects replication, placement, execution, and self-corretion		X	FR/RM PR/RE	[++]	[++]	[++]	
Resilient cloud collaboration for resource sharing	Cloud site and multiple clouds	Content placement and run-time	Cloud brokering, content dissemination, and storage		X	PR/RE RM/FR	+++	[+++]	++	+

Type and usage refer to main categorization of resiliency techniques, and their domain and expected usage in cloud computing.
Methods and goals refer to three aspects, which are the main methods, main goals besides survivability, and resiliency strategies adopted by approaches, techniques, and architecture of each main category. The main goals besides survivability are the minimization of recovery time (RT) and / or minimization of cost. The resiliency strategies are protection "PR", recovery "RE", removal "RM", and forecasting "FR" (Section IV-B). Methodologies also are, "VA" vulnerability aware, "CD" capacity dimensioning, "Place" content/data center placement, "AnyCast" routing, "DR" disaster resiliency, "VBN" virtual backup node mapping.
Survivability enforced consists in the level of resiliency which is expected to be achieved by each category or approach (similar to Tables III and IV) Note that "[ ]" implies that the level of resiliency enforced might depends on the implementation of the technique and type of failure scenarios.
∗Survivability in different levels of cloud computing application failures, "+" minor failure in process and data, "++" failure of multiple process and data (i.e., cloud application or service unreachable temporary), "+++" failure in the whole application and site (i.e., cloud application or service corruption with losses),
†Survivability in different level of attacks, "+" attacks on one tenant and minor services disruption "++" attacks affecting a DC with some data loses in more than one tenant, difficult to mitigate, "+++" sophisticated attack with important data losses in several tenants, when the attacker might control some important equipment or servers.
$^{\epsilon}$Survivability in different level of cloud infrastructure failures, "+" failure of single component of the DC including one or more servers and communication devices in the intra/inter DCN, "++" failure of one DC and/or links of the inter DCN, "+++" failure of many links and components of the inter-DCN, and more than one DCs are disconnected or an entire cloud infrastructure.
†† Survivability in case of disaster and post-disaster cascading failures, "+" one disaster with low risk of cascading failure, "++" more severe disaster with moderate risk of cascading failures, "+++" large disaster with high risk of cascading failure.

protecting against single (physical) server failures (in terms of both software and hardware). Designers and operators typically add resiliency functionalities for operating systems (OSs), physical machines (PMs), and storage to enforce basic resiliency. A higher degree of resiliency, *mid*, is reached through server virtualization: introducing the abstraction of virtual machines (VMs) enables VM replication, migration, placement, and checkpointing techniques. However, both physical and virtual server resiliency techniques rely on the network within and among data centers (i.e., intra- and inter-DCN) for transferring and/or updating the VM replicas. These networks can be virtualized as well, or not, with specific techniques to make them resilient.

The main method for resiliency of *non-virtualized* networks is redundancy. At the (physical) topology level, this may be at the switch or port level, and may also involve augmentation of the topology to improve its resiliency. Looking at the actual routes followed by the cloud traffic (which may span multiple physical hops), resiliency approaches comprise (i) (re)routing traffic flows to decide what route to use (possibly to minimize failures, or rerouting once the original path is affected), and (ii) provisioning capacity (and possibly extending the topology) of the network links such that it also suffices in case of failures (i.e., accounts for possibly rerouted traffic). The first amounts to run-time decisions, while the latter can also impact the planning stage. Virtualization implies that the mapping of the virtual network (VN) onto the physical infrastructure needs to be "planned", which amounts to virtual network mapping (which, from the perspective of the cloud infrastructure provider, is also a run-time process, albeit on larger time scales than routing of traffic over the virtual traffic). Adding virtualization amounts to multi-layer resiliency, opening up possibilities for, e.g., cross-layer backup strategies. Another important concept is anycast routing which can be exploited to reroute to different data centers, depending on the failures.

Resiliency techniques for networks and servers achieve a *high* level of protection against physical failures, *basic* to *high* level against software failures, and *basic* in case of disasters. The most resilient techniques for servers are VM placement and checkpointing. For network virtualization, the higher resiliency is achieved by VN mapping with cross-layer resiliency approaches considering cut-disjoint VN mapping. Software-defined networking (SDN) resiliency provides a higher level of disaster resiliency, given the ability to self-replicate and place the central controller to avoid disaster failures in the physical network.

In conclusion, isolated resiliency techniques for facility, servers, and networks can ensure only partial resiliency of the cloud infrastructure (i.e, processing, storage, or communication separately), and they target single or a reduced number of physical or software failures. For large failures (e.g., disasters) and complex cascading failures, a *high* level of resiliency requires coordination of the network and computing part of system, as discussed below.

2. *Resiliency in cloud integrated and middleware infrastructure* integrates resilient techniques for facilities, servers, and networks to achieve higher levels of resiliency (or do so more efficiently) in complex failure scenarios compared to separate strategies for the network/server parts. Most strategies pertain to cloud run-time and planning processes, with some exceptions of architecture design. The most common strategies for integrated non-virtualized cloud infrastructure are: (i) disaster and network vulnerability-aware data center/content placement and connectivity, (ii) anycast and capacity dimensioning for content placement and dissemination, and (iii) data/content replication and evacuation techniques.

Cloud networks (CN) exploit virtualization in cloud integrated infrastructure to provide more flexibility and resiliency. Cloud architecture design has been proposed to allow the integration of virtual networks (VNs) and virtual machines (VMs) in cloud networks by enabling resiliency capabilities for VMs and VNs. Resiliency techniques for cloud network mapping combine techniques for virtualized networks and servers resiliency such as: backup-facility node mapping, full VM or virtual node replication, anycast access to replicated content, content connectivity guarantee, and data evacuation.

Resiliency techniques for cloud networks build on resiliency approaches borrowed from non-integrated network infrastructure, without virtualization. However, virtualization and appropriate CN mapping approaches change the picture, as multiple resiliency techniques can be combined to achieve a *high* level of resiliency against software and hardware failures, and *mid* level in disasters. The only exception again is Byzantine attack which depends on the level of awareness used by each approach to detect and relocate VMs or contents affected by the attacks.

Finally, several resiliency approaches have been proposed for implementation in the middleware sub-layer: checkpoint orchestration, resilient connectivity (TPC/IP and SDN), resilient scheduling and load balancing, middleware architectures that build mobile targets for attacks, and service-oriented middleware architectures.

3. *Cloud application resiliency* are identified in Table VI, including the following major trends of surveyed approaches for cloud application design and operation:

3.a) *Cloud application design and development approaches* tend to provide *basic* and *mid* levels of application survivability for the case of software failures and Byzantine attacks. For hardware failures, application design and development achieve just *basic* levels, given the severe impact of physical failures on the cloud application.

3.b) *Cloud collaboration for resources sharing and data/content management approaches* achieve a higher level of cloud application resiliency compared to application design and development approaches, because they exploit inter-cloud collaboration, resource brokering, and content dissemination methods to replicate, allocate, and relocate cloud applications. Most approaches achieve *mid* level survivability for disaster failures, however none of them approach support for cascading failures.

B. Final remarks and open challenges

The most common resiliency techniques used in cloud system rely on data replication and checkpointing from the storage side, virtualization and migration from the computing side, and multi-layer protection and anycast routing from the networking side. Given the growing importance of cloud systems in our daily life, integrated approaches combining such features, especially to deal with extreme scenarios such as disasters or intentional attacks, are emerging and represent an important research direction for the future years.

Open areas in cloud infrastructure and application resiliency are the following:

1. The aforementioned *integration and coordination between the different approaches* for multiple and complex failures (e.g., disaster and post-disaster failures) is an open challenge for the "vertical" integration of approaches for cloud-integrated infrastructures, cloud middleware architectures, and cloud applications, due to the high computational complexity required to optimize a very diverse cloud infrastructure (i.e., federated clouds). A key enabler for this integration can be the logical centralization of the vertically-integrated control (as in the SDN paradigm). However, the optimal control placement and replication are open research problems.

2. *Scalability of (algorithms for) resiliency*, as current optimization tools, models, and approaches for cloud systems are often limited to small problem sizes, but such systems are continuously growing in size and complexity. In particular, large coordination problems such as resource brokering and recovery scheduling (e.g., re-scheduling after disaster events) are expected to face scalability issues. The exploration of efficient machine learning based approaches and big data are also an open research area.

3. *Multi-domain extensions* to handle cooperative, self-adaptable, and geo-distributed recovery and protection in inter-cloud infrastructures (e.g., multi-domain emergency evacuation and recovery).

4. *Resiliency in multi-domain and multi-layer network function virtualization (NFV)* [232] to deal with the dynamicity of cloud computing which requires an efficient and resilient orchestration of multiple placement and interconnection of virtualized functions in one cloud provider or in a federation of clouds (e.g., multi-domain resiliency of multiple cloud service chains).

5. *Interdependency between different infrastructures due to cascading failures*, especially between the power grid and cloud infrastructure components [233].

6. *Mobile and access resiliency for cloud computing services*, given that the increase of access to cloud computing services from mobile devices will require resiliency for the case of disruption in 4G/5G mobile access networks [234].

7. *Energy-efficient resiliency techniques*, as resiliency techniques typically lead to increased energy consumption in cloud infrastructures.

8. *Risk engineering* in complex and multi-domain cloud infrastructure to improve the preparedness and flexibility of cloud providers to the uncertainty of failures.

9. *Resilient programmability of multiple cloud layers and domains* to extend and to integrate existing proposed approaches using SDN and programmable metro and intra-DCNs (i.e., Software Defined Optical Datacenter Networks (SDO-DN)).

VIII. Glossary of Terms

- *Backup route existence* is a topology property requiring that every source node can reach any destination node after a failure (e.g., disconnected node or broken link).
- *Business continuity plan* is a set of procedures designed to reduce the impact of large failures over service operations.
- *Byzantine attack/failure* is an abnormal behavior of a compromised (i.e., controlled by an attacker) VM or VS server that produces Denial-of-Service (DOS) attacks on other services or components of the same server and cloud infrastructure that might produce important failures.
- *Cloud Network (CN)* is a set of VMs connected using a VN, assigned for tenants virtual site (VDS) and/or cloud application, which also can be used as *virtual infrastructure (VI)* [235] or as *virtual data centers (VDC)* [136]).
- *Critical cut* complies a set of links or nodes which, when failing simultaneously, divide the topology in disconnected parts.
- *A Data center (DC)* facility is a specialized location for servers or physical machines (PM).
- *High nodal degree* indicates a high average number of links connected to each node (which typically implies that critical cuts will contain many links).
- *Migration* comprises the relocation of services, data, or applications from one physical location to another for resiliency or cost purposes.
- *A Physical Machine (PM)* consists of a specialized computer with multiple processors and storage capabilities (i.e., attached storage and network access storage (NAS)).
- *Proof of Retrievability (POR)* scheme allows the backup service (the "prover") to produce a concise proof that a user (the "verifier") can recover a specific file or data content without corruption and interruption [236].
- *Resiliency* is the capacity of a system or an infrastructure or a business to remain reliable (i.e., reliability), dependable (i.e., dependability), failure tolerant, survivable (i.e., recoverable) and secure (i.e., incorruptible) in case of any malicious or accidental malfunctions or failures that result in a temporal or permanent service disruption. *Reliability* is the probability that a system remains operative in a period of time. *Failure tolerance* is the system's level of tolerance for a given failure. *Survivability* is the capability of a system to survive (i.e., recover) a certain kind of failures (i.e., single, multiple, and disasters) and attacks (i.e., WMD and Byzantine). *Dependability* combines reliability, failure tolerance, and

security. Additionally, survivability refers to multiple dependability towards a diverse type of failures. *Security* means data and/or communication integrity, confidentiality, and reliability. Resiliency includes response to failures or malfunctions caused by security violation (e.g., Byzantine attacks/failures) [8], [9], [10].

- *Service-level objective (SLO)* is the level of quality of service (QoS) enforced by the service provider based on the service-level agreement (SLA) with the tenant and/or end users.
- *A Tenant* is a user who rents a fraction of the cloud infrastructure capacity to provide cloud application services for end users.
- *A Virtual Machine (VM)* is the virtualization of computing and storage on a physical server (i.e., virtual memory (RAM), virtual hard disk, and virtual processor by using a virtualized operating system).
- *A Virtual Digital Site (VDS)* is a set of VMs assigned to a specific tenant that is considered as single system.
- *A Virtual Network (VNet/VN)* represent a set of virtual nodes (i.e., virtual routers) connected by virtual links allocated (VN mapping) in a shared physical network for a given tenant or service.
- *VN mapping (a.k.a. embedding)* consists in the allocation of resources for a tenant's VN in nodes and links of a physical network (i.e., substrate network).

ACKNOWLEDGMENT

We grateful thank the Editor and Reviewers for their constructive criticisms which significantly improved the value of this paper. Carlos Colman-Meixner's work as part of his PhD studies has been supported by a Fulbright / Itaipu Scholarship. Part of this work was also supported by the U.S. Defense Threat Reduction Agency (DTRA).

REFERENCES

[1] D. F. Parkhill, *The challenge of the computer utility*. Addison-Wesley Professional, USA, 1966.

[2] M. Hamdaqa and L. Tahvildari, "Cloud computing uncovered: A research landscape," *Springer, Advances in Computers*, vol. 86, pp. 41 – 85, July 2012.

[3] A. Mohamed, "A history of cloud computing," *Computer Weekly*, March 2009.

[4] P. M. Mell and T. Grance, "SP 800-145. the NIST definition of cloud computing," Tech. Rep., Oct. 2011.

[5] L. Colombus, "Predicting Enterprise Cloud Computing Growth, Forbes, Sept. 2013." [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2013/09/04/predicting-enterprise-cloud-computing-growth/

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, April 2010.

[7] C. Cerin, C. Coti, P. Delort, and F. Diaz, "Downtime statistics of current cloud solutions," Tech. Rep., June 2013.

[8] M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein, "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 2, pp. 106–124, June 2009.

[9] V. Panagiotis, S. Vera, D. Panagiotis, C. Scott, G. Slawomir, and I. Demosthenes, "Ontology and taxonomies of resilience," European Network and Information Security Agency (ENISA), Tech. Rep. V1, Dec. 2011.

[10] NIAC, "Critical infrastructure resilience, final report and recommendations," USA National Infrastructure Advisory Concil (NIAC)- Homeland Security, Tech. Rep. V1, Sep. 2009.

[11] P. Cholda, A. Mykkeltveit, B. Helvik, O. Wittner, and A. Jajszczyk, "A survey of resilience differentiation frameworks in communication networks," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 4, pp. 32–55, Dec. 2007.

[12] M. F. Habib, M. Tornatore, F. Dikbiyik, and B. Mukherjee, "Disaster survivability in optical communication networks," *Elsevier, Computer Communications*, vol. 36, no. 6, pp. 630 – 644, March 2013.

[13] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proc. ACM SIGCOMM'11 conference*, Aug. 2011, pp. 350–361.

[14] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 431–442, Oct. 2012.

[15] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Springer, Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010.

[16] I. Abbadi, "Clouds infrastructure taxonomy, properties, and management services," *Springer, Advances in Computing and Communications*, vol. 193, pp. 406–420, July 2011.

[17] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb. 2009.

[18] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, May 2013.

[19] B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. Fifth International Joint Conference on INC, IMS and IDC, NCM'09*, Aug. 2009, pp. 44–51.

[20] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *Proc. 2nd International Conference on Computer and Network Technology (ICCNT'10)*, April 2010, pp. 222–226.

[21] S. Rajagopalan, B. Cully, R. O'Connor, and A. Warfield, "SecondSite: disaster tolerance as a service," *ACM SIGPLAN Notice - VEE'12*, vol. 47, no. 7, pp. 97–108, July 2012.

[22] A. Gohad, K. Ponnalagu, N. Narendra, and P. Rao, "Towards self-adaptive cloud collaborations," in *Proc. IEEE International Conference on Cloud Engineering (IC2E)*, March 2013, pp. 54–61.

[23] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: dependable and secure storage in a cloud-of-clouds," in *Proc. 6th ACM Conference on Computer Systems*, ser. EuroSys '11, April 2011, pp. 31–46.

[24] X. Wu, D. Turner, C.-C. Chen, D. A. Maltz, X. Yang, L. Yuan, and M. Zhang, "NetPilot: automating datacenter network failure mitigation," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 419–430, Aug. 2012.

[25] C. Hoff and P. Simmons, "Security guidance for critical areas of focus in cloud computing, domain 1," Cloud Security Alliance, Tech. Rep., Nov. 2011.

[26] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Oct. 2004.

[27] M. Grottke, R. Matias, and K. Trivedi, "The fundamentals of software aging," in *Proc. IEEE International Conference on Software Reliability Engineering Workshops*, Nov. 2008.

[28] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proc. 1st ACM symposium on Cloud Computing (SoCC'10)*, June 2010, pp. 193–204.

[29] CA Technologies, "Research report, the avoidable cost of downtime," Tech. Rep., Nov. 2010. [Online]. Available: http://m.softchoice.com/files/pdf/brands/ca/ACOD_REPORT.pdf

[30] T. Wood, H. A. Lagar-Cavilla, K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "PipeCloud: using causality to overcome speed-of-light delays in cloud-based disaster recovery," in *Proc. 2nd ACM Symposium on Cloud Computing (SOCC'11)*, Oct. 2011, pp. 17:1–17:13.

[31] D. Sun, G. Chang, C. Miao, and X. Wang, "Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments," *Springer, The Journal of Supercomputing*, vol. 66, no. 1, pp. 193–228, March 2013.

[32] L. Chen and A. Avizienis, "N-version programming: A fault-tolerance approach to rellablllty of software operatlon," in *Proc. 25th International Symposium on Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years*, June 1995.

[33] B. Randell and J. Xu, *The evolution of recovery block concept - Software Fault Tolerance*. John Wiley Ltd, 1994.

[34] P. Chan, M. Lyu, and M. Malek, "ReliableWeb services: Methodology, experiment and modeling," in *Proc. IEEE International Conference on Web Services, (ICWS'07)*, July 2007, pp. 679–686.

[35] J. Panneerselvam, L. Liu, R. Hill, Y. Zhan, and W. Liu, "An investigation of the effect of cloud computing on network management," in *Proc. IEEE 9th International Conference on High Performance Computing and Communication, IEEE 14th International Conference on Embedded Software and Systems (HPCC-ICESS'12)*, June 2012, pp. 1794–1799.

[36] T. W.Pitt, J. H. Seader, and V. E. Renaud, "Datacenter site infrastructure tier standard:topology," *Uptime Institute, The Global Datacenter Authority*, Tech. Rep., 2010.

[37] S. Ruj and A. Pal, "Analyzing cascading failures in smart grids under random and targeted attacks," in *Proc. IEEE 28th International Conference on Advanced Information Networking and Applications (AINA'14)*, May 2014, pp. 226–233.

[38] S. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, no. 7291, pp. 1025–1028, 2007.

[39] A. Fukui, T. Takeda, K. Hirose, and M. Yamasaki, "HVDC power distribution systems for telecom sites and data centers," in *Proc. International Power Electronics Conference (IPEC'10)*, June 2010, pp. 874–880.

[40] R. Aldrich and G. Mellinger, "Cisco energy management: A case study in implementing energy as a service," July 2009, white paper. [Online]. Available: http://www.cisco.com/c/dam/en/us/products/collateral/switches/energywise-technology/CiscoEMSWhitePaper.pdf

[41] P. M. Curtis, *Maintaining Mission Critical Systems in a 24/7 Environment, 2nd Edition*. Wiley-IEEE Press, 2011.

[42] A. Sedgewick, "Framework for improving critical infrastructure cybersecurity, version 1.0," Tech. Rep., Feb. 2014.

[43] N. Carter, H. Naeimi, and D. Gardner, "Design techniques for cross-layer resilience," in *Proc. IEEE Design, Automation Test in Europe Conference Exhibition (DATE'10)*, March 2010, pp. 1023–1028.

[44] A. Shye, J. Blomstedt, T. Moseley, V. Reddi, and D. Connors, "PLR: A software approach to transient fault tolerance for multicore architectures," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 2, pp. 135–148, April 2009.

[45] I. Egwutuoha, S. Chen, D. Levy, and B. Selic, "A fault tolerance framework for high performance computing in cloud," in *Proc. 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*, May 2012, pp. 709–710.

[46] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith, "Configurable isolation: Building high availability systems with commodity multi-core processors," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 470–481, June 2007.

[47] A. Tchana, L. Broto, and D. Hagimont, "Approaches to cloud computing fault tolerance," in *Proc. International Conference on Computer, Information and Telecommunication Systems (CITS'12)*, May 2012.

[48] A. Saleh, J. Serrano, and J. Patel, "Reliability of scrubbing recovery-techniques for memory systems," *IEEE Transactions on Reliability*, vol. 39, no. 1, pp. 114–122, April 1990.

[49] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[50] S. B. Wicker, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: IEEE Press, 1994.

[51] A. Leventhal, "Triple-parity RAID and beyond," *Communications of the ACM*, vol. 53, no. 1, pp. 58–63, Jan. 2010.

[52] S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The design and implementation of Zap: A system for migrating computing environments," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 361–376, Dec. 2002.

[53] S. Potter and J. Nieh, "AutoPod: Unscheduled system updates with zero data loss," in *Proc. IEEE Second International Conference on Autonomic Computing, (ICAC'05)*, June 2005, pp. 367–368.

[54] D. J. Scales, M. Nelson, and G. Venkitachalam, "The design of a practical system for fault-tolerant virtual machines," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 4, pp. 30–39, Dec. 2010.

[55] 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," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, Oct. 2003.

[56] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proc. USENIX Annual Technical Conference (ATEC'05)*, Apr. 2005.

[57] M. R. Hines, U. Deshpande, and K. Gopalan, "Post-copy live migration of virtual machines," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 3, pp. 14–26, July 2009.

[58] G. Vallee, C. Engelmann, A. Tikotekar, T. Naughton, K. Charoenpornwattana, C. Leangsuksun, and S. Scott, "A framework for proactive fault tolerance," in *Proc. Third International Conference on Availability, Reliability and Security (ARES'08)*, March 2008, pp. 659–664.

[59] S. Fu, "Failure-aware construction and reconfiguration of distributed virtual machines for high availability computing," in *Proc. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, (CCGRID '09)*, May 2009, pp. 372–379.

[60] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement," in *Proc. IEEE International Conference on Services Computing (SCC)*, 2011, pp. 72–79.

[61] S. Malik, F. Huet, and D. Caromel, "RACS: A framework for resource aware cloud computing," in *Proc. International Conference For Internet Technology and Secured Transactions, (ICITST'12)*, Dec. 2012, pp. 680–687.

[62] E. Bin, O. Biran, O. Boni, E. Hadad, E. Kolodner, Y. Moatti, and D. Lorenz, "Guaranteeing high availability goals for virtual machine placement," in *Proc. 31st International Conference on Distributed Computing Systems (ICDCS'11)*, June 2011, pp. 700–709.

[63] Q. Zhang, M. Li, and X. Hu, "Network traffic-aware virtual machine placement with availability guarantees based on shadows," in *Proc. 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'14)*, May 2014, pp. 542–543.

[64] J. J. Moore and C. Kesselman, "A resiliency model for high performance infrastructure based on logical encapsulation," in *Proc. 21st ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC '12)*, July 2012, pp. 283–294.

[65] M. Kozuch and M. Satyanarayanan, "Internet suspend/resume," in *Proc. 4th IEEE Workshop on Mobile Computing Systems and Applications 2002*, 2002, pp. 40–46.

[66] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *Proc. 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, April 2008, pp. 161–174.

[67] V. Medina and J. M. García, "A survey of migration mechanisms of virtual machines," *ACM Computing Surveys*, vol. 46, no. 3, pp. 30:1–30:33, Jan. 2014.

[68] E. Park, B. Egger, and J. Lee, "Fast and space-efficient virtual machine checkpointing," in *Proc. 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'11)*, March 2011, pp. 75–86.

[69] U. F. Minhas, S. Rajagopalan, B. Cully, A. Aboulnaga, K. Salem, and A. Warfield, "RemusDB: transparent high availability for database systems," *Springer, The VLDB Journal*, vol. 22, no. 1, pp. 29–45, Feb. 2013.

[70] L. B. Gomez, B. Nicolae, N. Maruyama, F. Cappello, and S. Matsuoka, "Scalable reed-solomon-based reliable local storage for HPC applications on iaas clouds," in *Proc. 18th International Conference on Parallel Processing Workshop (Euro-Par'12)*. Springer-Verlag, Aug. 2012, pp. 313–324.

[71] Y. Han, R. Zheng, and W. H. Mow, "Exact regenerating codes for Byzantine fault tolerance in distributed storage," in *Proc. 31st IEEE International Conference on Computer Communication (INFOCOM'12)*, March 2012, pp. 2498–2506.

[72] D. Dobre, G. Karame, W. Li, M. Majuntke, N. Suri, and M. Vukolić, "Powerstore: Proofs of writing for efficient and robust storage," in *Proc.*

ACM SIGSAC Conference on Computer and Communications Security (CCS'13), Nov. 2013, pp. 285–298.

[73] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in *Proc, 16th ACM SIGSAC International Conference on Computer and Communications Security (CCS'09)*, Nov. 2009, pp. 187–198.

[74] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, "Depot: Cloud storage with minimal trust," *ACM Transaction on Computer Systems*, vol. 29, no. 4, pp. 12:1–12:38, Dec. 2011.

[75] R. Boutaba, Q. Zhang, and M. F. Zhani, "Virtual machine migration: Benefits, challenges and approaches," *Communication Infrastructures for Cloud Computing: Design and Applications*, pp. 383–408, Sept. 2013.

[76] K.-Y. Hou, K. G. Shin, Y. Turner, and S. Singhal, "Tradeoffs in compressing virtual machine checkpoints," in *Proc. 7th International ACM Workshop on Virtualization Technologies in Distributed Computing (VTDC'13)*, June 2013, pp. 41–48.

[77] F. Lopez Pires and B. Baran, "A virtual machine placement taxonomy," in *Proc. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'15)*, May 2015, pp. 159–168.

[78] S. Fu, "Failure-aware resource management for high-availability computing clusters with distributed virtual machines," *Elsevier, Journal of Parallel Distributed Computing*, vol. 70, no. 4, pp. 384–393, April 2010.

[79] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[80] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," *Communications of the ACM*, vol. 54, no. 3, pp. 95–104, March 2011.

[81] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Elsevier, Computer Communications*, vol. 40, pp. 1 – 21, March 2014.

[82] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM Conference (SIGCOMM'08)*, August 2008, pp. 75–86.

[83] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," *SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, Aug. 2009.

[84] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: Using backup port for server interconnection in data centers," in *Proc. 28th IEEE International Conference in Computer Communication (INFOCOM'09)*, April 2009, pp. 2276–2285.

[85] Y. Liao, J. Yin, D. Yin, and L. Gao, "DPillar: Dual-port server interconnection network for large scale data centers," *Elsevier, Computer Networks*, vol. 56, no. 8, pp. 2132 – 2147, May 2012.

[86] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proc. 2nd ACM Symposium on Cloud Computing (SOCC'11)*, Oct. 2011, pp. 2:1–2:13.

[87] L. Gyarmati and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 5, pp. 4–12, Oct. 2010.

[88] L. Gyarmati, A. Gulys, B. Sonkoly, T. A. Trinh, and G. Biczk, "Free-scaling your data center," *Elsevier, Computer Networks*, vol. 57, no. 8, pp. 1758 – 1773, June 2013.

[89] W. Ni, C. Huang, and J. Wu, "Provisioning high-availability datacenter networks for full bandwidth communication," *Elsevier, Computer Networks*, vol. 68, pp. 71 – 94, Aug. 2014.

[90] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks. part i-protection," in *Proc. 18th IEEE International Conference of Computer Communication (INFOCOM'99)*, Mar 1999, pp. 744–751.

[91] C. S. Ou, J. Zhang, H. Zang, L. H. Sahasrabuddhe, and B. Mukherjee, "New and improved approaches for shared-path protection in wdm mesh networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 22, no. 5, pp. 1223–1232, May 2004.

[92] R. Asthana, Y. Singh, and W. Grover, "p-Cycles: An overview," *IEEE Communications Surveys and Tutorials*, vol. 12, no. 1, pp. 97–111, Feb. 2010.

[93] M. Kiaei, C. Assi, and B. Jaumard, "A survey on the p-Cycle protection method," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 3, pp. 53–70, Aug. 2009.

[94] S. Sebbah and B. Jaumard, "A resilient transparent optical network design with a pre-configured extended-tree scheme," in *Proc. IEEE International Conference on Communications, (ICC'09)*, June 2009.

[95] C. Ma, J. Zhang, Y. Zhao, J. Yuan, Z. Jin, X. Li, S. Huang, and W. Gu, "Pre-configured prism (p-Prism) method against simultaneous dual-link failure in optical networks," *Elsevier, Optical Fiber Technology*, vol. 20, no. 5, pp. 443 – 452, Oct. 2014.

[96] F. Dikbiyik, M. D. Leenheer, A. Reaz, and B. Mukherjee, "Minimizing the disaster risk in optical telecom networks," in *Proc. IEEE/OSA Optical Fiber Communication Conference (OFC'12)*, March 2012.

[97] F. Dikbiyik, M. Tornatore, and B. Mukherjee, "Minimizing the risk from disaster failures in optical backbone networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 32, no. 18, pp. 3175–3183, Sept. 2014.

[98] E. Modiano and A. Narula-Tam, "Survivable routing of logical topologies in WDM networks," in *Proc. 20th IEEE International Conference on Computer Communication (INFOCOM'01)*, 2001, pp. 348–357.

[99] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: A survey," *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11–20, Jan. 2012.

[100] A. Todimala and B. Ramamurthy, "A scalable approach for survivable virtual topology routing in optical WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 63–69, August 2007.

[101] C. Liu and L. Ruan, "A new survivable mapping problem in IP-over-WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 3, pp. 25–34, April 2007.

[102] M. Bui, B. Jaumard, C. Cavdar, and B. Mukherjee, "Design of a survivable VPN topology over a service provider network," in *Proc. 9th International Conference on the Design of Reliable Communication Networks (DRCN'13)*, March 2013, pp. 71–78.

[103] K. Thulasiraman, M. Javed, T. Lin, and G. Xue, "Logical topology augmentation for guaranteed survivability under multiple failures in IP-over-WDM optical network," in *Proc. 3rd IEEE International Symposium on Advanced Networks and Telecommunication Systems (ANTS'09)*, Dec. 2009.

[104] D.-J. Kan, A. Narula-Tam, and E. Modiano, "Lightpath routing and capacity assignment for survivable IP-over-WDM networks," in *Proc. 7th International Workshop on Design of Reliable Communication Networks, (DRCN'09)*, Oct. 2009, pp. 37–44.

[105] M. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, June 2013.

[106] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared backup network provision for virtual network embedding," in *Proc. IEEE International Conference on Communications (ICC'11)*, June 2011.

[107] C. S. Vadrevu and M. Tornatore, "Survivable IP topology design with re-use of backup wavelength capacity in optical backbone networks," *Elsevier, Optical Switching and Networking*, vol. 7, no. 4, pp. 196 – 205, Dec. 2010.

[108] J. Rak, "Fast service recovery under shared protection in WDM networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 30, no. 1, pp. 84–95, Jan. 2012.

[109] J. Shamsi and M. Brockmeyer, "QoSMap: Achieving quality and resilience through overlay construction," in *Proc. 4th. International Conference on Internet and Web Applications and Services, (ICIW'09)*, May 2009, pp. 58–67.

[110] X. Zhang, C. Phillips, and X. Chen, "An overlay mapping model for achieving enhanced QoS and resilience performance," in *Proc. 3rd. International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT'11)*, Oct. 2011.

[111] X. Zhang and C. Phillips, "A novel heuristic for overlay mapping with enhanced resilience and QoS," in *Proc. IET International Conference on Communication Technology and Application (ICCTA'11)*, Oct. 2011, pp. 540–545.

[112] A. Bagula, "Improving the resilience of emerging generation GMPLS networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 1, no. 2, pp. A56–A68, June 2009.

[113] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient openflow-based networking," in *Proc. IEEE Network Operations and Management Symposium (NOMS'12)*, April 2012, pp. 933–939.

[114] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in *Proc. 22nd International Conference on Computer Communications and Networks (ICCCN'13)*, July 2013.

[115] J. P. Sterbenz, D. Hutchison, E. K. Cetinkaya, A. Jabbar, J. P. Rohrer, M. Schller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Elsevier, Computer Networks*, vol. 54, no. 8, pp. 1245 – 1265, June 2010.

[116] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Elsevier, Computer Networks*, vol. 54, no. 5, pp. 862–876, Apr. 2010.

[117] Y. Zhang, A.-J. Su, and G. Jiang, "Understanding data center network architectures in virtualized environments: A view from multi-tier applications," *Elsevier, Computer Networks*, vol. 55, no. 9, pp. 2196 – 2208, June 2011.

[118] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. USA: Morgan Kaufmann Publishers Inc., 2004.

[119] B. Mukherjee, *Optical WDM Networks*. Springer, 2006.

[120] W.-H. Liao and S.-C. Su, "A dynamic VPN architecture for private cloud computing," in *Proc. 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*, Dec. 2011, pp. 409–414.

[121] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, Nov. 2013.

[122] A. Valdivieso Caraguay, L. Barona Lopez, and L. Garcia Villalba, "Evolution and challenges of software defined networking," in *Proc. IEEE SDN for Future Networks and Services (SDN4FNS'13)*, Nov. 2013, pp. 1–7.

[123] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of fiber infrastructure to disasters," *IEEE/ACM Transactions in Networking*, vol. 19, no. 6, pp. 1610–1623, Dec. 2011.

[124] B. Jaumard, A. Hoang, and M. Bui, "Path vs. cutset approaches for the design of logical survivable topologies," in *Proc. IEEE International Conference on Communications (ICC'12)*, June 2012, pp. 3061–3066.

[125] N. Kamiyama, "Designing data center network by analytic hierarchy process," *Elsevier, Computer Networks*, vol. 57, no. 3, pp. 658 – 667, Feb. 2013.

[126] S. Ferdousi, F. Dikhiyik, M. Habib, and B. Mukherjee, "Disaster-aware data-center and content placement in cloud networks," in *Proc. IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS'13)*, Dec. 2013.

[127] S. Ferdousi, M. Tornatore, B. Mukherjee, M. F. Habib, and F. Dikbiyik, "Disaster-aware dynamic content placement in optical cloud networks," in *Proc. IEEE/OSA Optical Fiber Communication Conference (OFC'14)*, March 2014, p. M2H.4.

[128] B. Jaumard, A. Shaikh, and C. Develder, "Selecting the best locations for data centers in resilient optical grid/cloud dimensioning," in *Proc. 14th International Conference on Transparent Optical Networks (ICTON'12)*, 2012.

[129] S. S. Savas, F. Dikbiyik, M. F. Habib, M. Tornatore, and B. Mukherjee, "Disaster-aware service provisioning with manycasting in cloud networks," *Springer, Photonic Network Communications*, vol. 28, no. 2, pp. 123–134, Oct. 2014.

[130] C. Develder, J. Buysse, M. De Leenheer, B. Jaumard, and B. Dhoedt, "Resilient network dimensioning for optical grid/clouds using relocation," in *Proc. IEEE International Conference on Communications (ICC'12)*, 2012, pp. 6262–6267.

[131] C. Develder, J. Buysse, B. Dhoedt, and B. Jaumard, "Joint dimensioning of server and network infrastructure for resilient optical grids/clouds," *IEEE/ACM Transactions on Networking*, vol. 22, no. 5, pp. 1591–1606, Oct 2014.

[132] M. F. Habib, M. Tornatore, M. De Leenheer, F. Dikbiyik, and B. Mukherjee, "Design of disaster-resilient optical datacenter networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 30, no. 16, pp. 2563–2573, Aug. 2012.

[133] J. Yao, P. Lu, and Z. Zhu, "Minimizing disaster backup window for geo-distributed multi-datacenter cloud systems," in *IEEE International Conference on Communications (ICC'14)*, June 2014, pp. 3631–3635.

[134] S. Ferdousi, M. F. Habib, M. Tornatore, and B. Mukherjee, "Rapid data evacuation for large-scale disasters in optical cloud networks," in *Proc. IEEE/OSA Optical Fiber Communication Conference (OFC'15)*, March 2015, p. M3I.2.

[135] Q. Guan, Z. Zhang, and S. Fu, "Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems," in *Proc. 6th International Conference onAvailability, Reliability and Security (ARES'11)*, August 2011, pp. 83–90.

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

[137] C. Qiao, B. Guo, S. Huang, J. Wang, T. Wang, and W. Gu, "A novel two-step approach to surviving facility failures," in *Proc. IEEE/OSA Optical Fiber Communication Conference (OFC'11)*, March 2011.

[138] Q. Hu, Y. Wang, and X. Cao, "Location-constrained survivable network virtualization," in *Proc. 35th IEEE Sarnoff Symposium (SARNOFF'12)*, May 2012.

[139] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *Proc. IEEE International Conference on Communications (ICC'11)*, 2011.

[140] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'10)*, Dec. 2010.

[141] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *Proc. 5th IEEE International Conference on Cloud Computing (CLOUD'12)*, June 2012, pp. 196–203.

[142] V. Lira, E. Tavares, S. Fernandes, P. Maciel, and R. Silva, "Virtual network resource allocation considering dependability issues," in *Proc. 16th IEEE International Conference onComputational Science and Engineering (CSE'13)*, Dec. 2013, pp. 330–337.

[143] A. Kangarlou, P. Eugster, and D. Xu, "VNsnap: Taking snapshots of virtual networked environments with minimal downtime," in *Proc. IEEE/IFIP International Conference on Dependable Systems Networks, (DSN'09)*, June 2009, pp. 524–533.

[144] I. B. Barla, D. A. Schupke, and G. Carle, "Resilient virtual network design for end-to-end cloud services," in *Proc. IFIP TC 6 conference on Networking - Volume Part I (Networking'12)*, May 2012, pp. 161–174.

[145] M. Bui, B. Jaumard, and C. Develder, "Resilience options for provisioning anycast cloud services with virtual optical networks," in *Proc. IEEE International Conference on Communications (ICC'14)*, June 2014, pp. 3462–3468.

[146] Q. Zhang, M. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," in *Proc. 33rd IEEE International Conference on Computer Communication (INFOCOM'14)*, April 2014, pp. 289–297.

[147] H. Di, V. Anand, H. Yu, L. Li, D. Liao, and G. Sun, "Design of reliable virtual infrastructure using local protection," in *Proc. International Conference on Computing, Networking and Communications (ICNC'14)*, Feb. 2014, pp. 63–67.

[148] W.-L. Yeow, C. Westphal, and U. Kozat, "Designing and embedding reliable virtual infrastructures," in *Proc. 2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures,(VISA '10)*, pp. 33–40.

[149] G. A. S. Cavalcanti, R. R. Obelheiro, and G. Koslovski, "Optimal resource allocation for survivable virtual infrastructures," in *Proc. 10th International Conference on the Design of Reliable Communication Networks (DRCN'14)*, April 2014.

[150] H. Yu, C. Qiao, J. Wang, L. Li, V. Anand, and B. Wu, "Regional failure-resilient virtual infrastructure mapping in a federated computing and networking system," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 6, no. 11, pp. 997–1007, Nov. 2014.

[151] I. Barla, D. Schupke, M. Hoffmann, and G. Carle, "Optimal design of virtual networks for resilient cloud services," in *Proc. 9th International Conference on the Design of Reliable Communication Networks (DRCN'13)*, March 2013, pp. 218–225.

[152] M. F. Habib, M. Tornatore, and B. Mukherjee, "Fault-tolerant virtual network mapping to provide content connectivity in optical networks," in *Proc. IEEE/OSA Optical Fiber Communication Conference (OFC'13)*, March 2013, p. OTh3E.4.

[153] F. Gu, H. Alazemi, A. Rayes, and N. Ghani, "Survivable cloud networking services," in *Proc. IEEE International Conference on Computing, Networking and Communications (ICNC'13)*, Jan. 2013, pp. 1016–1020.

[154] F. Gu, K. Shaban, N. Ghani, S. Khan, M. Rahnamay-Naeini, M. Hayat, and C. Assi, "Survivable cloud network mapping for disaster recovery support," *IEEE Transactions on Computers*, vol. PP, no. 99, pp. 1–1, 2015.

[155] C. Colman-Meixner, F. Dikbiyik, M. Habib, M. Tornatore, C.-N. Chuah, and B. Mukherjee, "Disaster-survivable cloud-network mapping," *Springer, Photonic Network Communications*, vol. 27, no. 3, pp. 141–153, June 2014.

[156] C. Meixner, F. Dikbiyik, M. Tornatore, C. Chuah, and B. Mukherjee, "Disaster-resilient virtual-network mapping and adaptation in optical networks," in *Proc. 17th IFIP/IEEE International Conference on Optical Network Design and Modeling (ONDM'13)*, April 2013, pp. 107–112.

[157] X. Liu, C. Qiao, D. Yu, and T. Jiang, "Application-specific resource provisioning for wide-area distributed computing," *IEEE Network*, vol. 24, no. 4, pp. 25–34, July 2010.

[158] S. Savas, M. Habib, M. Tornatore, F. Dikbiyik, and B. Mukherjee, "Network adaptability to disaster disruptions by exploiting degraded-service tolerance," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 58–65, Dec. 2014.

[159] P. Ruth, X. Jiang, D. Xu, and S. Goasguen, "Virtual distributed environments in a shared infrastructure," *Computer*, vol. 38, no. 5, pp. 63–69, May 2005.

[160] M. Bui, B. Jaumard, and C. Develder, "Anycast end-to-end resilience for cloud services over virtual optical networks (invited)," in *Proc. 15th International Conference in Transparent Optical Network (ICTON'13)*, Cartagena, Spain, June 2013.

[161] E. Okorafor, "A fault-tolerant high performance cloud strategy for scientific computing," in *Proc. IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW'11)*, May 2011, pp. 1525–1532.

[162] C. Wang, L. Xing, H. Wang, Z. Zhang, and Y. Dai, "Processing time analysis of cloud services with retrying fault-tolerance technique," in *Proc. 1st IEEE International Conference on Communications in China (ICCC'12)*, August 2012, pp. 63–67.

[163] H. Jin, T. Ke, Y. Chen, and X.-H. Sun, "Checkpointing orchestration: Toward a scalable HPC fault-tolerant environment," in *Proc. 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*, May 2012, pp. 276–283.

[164] P. Das and P. Khilar, "VFT: A virtualization and fault tolerance approach for cloud computing," in *Proc. IEEE Conference on Information Communication Technologies (ICT'13)*, April 2013, pp. 473–478.

[165] B. Nicolae and F. Cappello, "BlobCR: Efficient checkpoint-restart for HPC applications on IaaS clouds using virtual disk image snapshots," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*, Nov. 2011.

[166] W. Zhao, P. Melliar-Smith, and L. Moser, "Fault tolerance middleware for cloud computing," in *Proc. 3rd IEEE International Conference on Cloud Computing (CLOUD'10)*, July 2010, pp. 67–74.

[167] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico Replication: A high availability framework for middleboxes," in *Proc. 4th ACM Symposium on Cloud Computing, (SOCC'13)*, 2013, pp. 1:1–1:15.

[168] P. Gupta, M. Goyal, and P. Kumar, "Trust and reliability based load balancing algorithm for cloud IaaS," in *Proc. 3rd IEEE International Advance Computing Conference (IACC'13)*, Feb. 2013, pp. 65–69.

[169] J. Jin, J. Luo, A. Song, F. Dong, and R. Xiong, "BAR: An efficient data locality driven task scheduling algorithm for cloud computing," in *Proc. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2011*, May 2011, pp. 295–304.

[170] S. Antony, A. Beegom, and M. Rajasree, "Task scheduling algorithm with fault tolerance for cloud," in *Proc. International Conference on Computing Sciences (ICCS'12)*, Nov. 2012, pp. 180–182.

[171] M. E. Frincu, "Scheduling highly available applications on cloud environments," *Future Generation Computer Systems*, vol. 32, pp. 138 – 153, 2014.

[172] J. Z. Li, Q. Lu, L. Zhu, L. Bass, X. Xu, S. Sakr, P. Bannerman, and A. Liu, "Improving availability of cloud-based applications through deployment choices," in *IEEE Sixth International Conference on Cloud Computing (CLOUD'13)*, June 2013, pp. 43–50.

[173] M. Jammal, A. Kanso, and A. Shami, "High availability-aware optimization digest for applications deployment in cloud," in *Proc. IEEE International Conference on Communications (ICC'15)*, June 2015, pp. 6822–6828.

[174] S. Malik, F. Huet, and D. Caromel, "Reliability aware scheduling in cloud computing," in *Proc. International Conferece For Internet Technology And Secured Transactions*, 2012, pp. 194–200.

[175] Q. Lu, X. Xu, L. Zhu, L. Bass, Z. Li, S. Sakr, P. Bannerman, and A. Liu, "Incorporating uncertainty into in-cloud application deployment decisions for availability," in *IEEE Sixth International Conference on Cloud Computing (CLOUD'13)*, June 2013, pp. 454–461.

[176] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE Systems Journal*, vol. 7, no. 2, pp. 288–297, June 2013.

[177] R. Jhawar and V. Piuri, "Fault tolerance management in IaaS clouds," in *Proc. First AESS European Conference on Satellite Telecommunications (ESTEL'12)*, 2012.

[178] A. Keromytis, R. Geambasu, S. Sethumadhavan, S. Stolfo, J. Yang, A. Benameur, M. Dacier, M. Elder, D. Kienzle, and A. Stavrou, "The MEERKATS cloud security architecture," in *Proc. IEEE 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW'12)*, June 2012, pp. 446–450.

[179] J. Baron, K. El Defrawy, A. Nogin, and R. Ostrovsky, "An architecture for a resilient cloud computing infrastructure," in *Proc. IEEE International Conference on Technologies for Homeland Security (HST'13)*, Nov. 2013, pp. 390–395.

[180] P. Verissimo, A. Bessani, and M. Pasin, "The TClouds architecture: Open and resilient cloud-of-clouds computing," in *Proc. IEEE/IFIP 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W'12)*, June 2012.

[181] Z. Zheng and M. R. Lyu, "An adaptive QoS-aware fault tolerance strategy for web services," *Springer, Empirical Software. Enggineeer*, vol. 15, no. 4, pp. 323–345, August 2010.

[182] Y. Wu and G. Huang, "Model-based high availability configuration framework for cloud," in *Proc. ACM Middleware Doctoral Symposium (MDS'13)*, Dec. 2013, pp. 6:1–6:6.

[183] Z. Zheng, T. Zhou, M. Lyu, and I. King, "FTCloud: A component ranking framework for fault-tolerant cloud applications," in *Proc. IEEE 21st International Symposium on Software Reliability Engineering (ISSRE'10)*, Nov. 2010, pp. 398–407.

[184] G. Fan, H. Yu, L. Chen, and D. Liu, "Model based Byzantine fault detection technique for cloud computing," in *Proc. IEEE Asia-Pacific Services Computing Conference (APSCS'12)*, Dec. 2012, pp. 249–256.

[185] G. Veronese, M. Correia, A. Bessani, L. C. Lung, and P. Verissimo, "Efficient Byzantine fault-tolerance," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 16–30, Jan 2013.

[186] J. Lau, L. C. Lung, J. da Fraga, and G. Santos Veronese, "Designing fault tolerant web services using BPEL," in *Proc. 7th IEEE/ACIS International Conference on Computer and Information Science (ICIS'08)*, May 2008, pp. 618–623.

[187] A. Benameur, N. Evans, and M. Elder, "Cloud resiliency and security via diversified replica execution and monioring," in *Proc. 6th International Symposium on Resilient Control Systems (ISRCS'13)*, Aug 2013, pp. 150–155.

[188] R.-H. Campbell, M. Montanari, and R. Farivar, "A middleware for assured clouds," *Springer, Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 87–94, Nov. 2012.

[189] M.-S. Bouguerra, D. Trystram, and F. Wagner, "Complexity analysis of checkpoint scheduling with variable costs," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1269–1275, June 2013.

[190] Vijindra and S. Shenai, "Survey on scheduling issues in cloud computing," *Elsevier, Procedia Engineering*, vol. 38, pp. 2881 – 2888, 2012.

[191] M. Jammal, A. Kanso, and A. Shami, "CHASE: Component high availability-aware scheduler in cloud computing environment," in *Proc. IEEE 8th International Conference on Cloud Computing (CLOUD'15)*, June 2015, pp. 477–484.

[192] H. S. Gunawi, T. Do, P. Joshi, P. Alvaro, J. M. Hellerstein, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, K. Sen, and D. Borthakur, "FATE and DESTINI: A framework for cloud recovery testing," in *Proc. 8th USENIX Conference on Networked Systems Design and Implementation, NSDI'11*, April 2011.

[193] P. Joshi, H. S. Gunawi, and K. Sen, "PREFAIL: a programmable tool for multiple-failure injection," *ACM SIGPLAN Notices*, vol. 46, no. 10, pp. 171–188, Oct. 2011.

[194] H. S. Gunawi, T. Do, J. M. Hellerstein, I. Stoica, D. Borthakur, and J. Robbins, "Failure as a Service (FaaS): A cloud service for large-scale, online failure drills," *EECS Department, University of California, Berkeley*, Tech. Rep. UCB/EECS-2011-87, July 2011.

[195] F. Faghri, S. Bazarbayev, M. Overholt, R. Farivar, R. H. Campbell, and W. H. Sanders, "Failure scenario as a service (FSaaS) for Hadoop clusters," in *Proc. ACM Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management (SDMCMM'12)*, Dec. 2012, pp. 5:1–5:6.

[196] Q. Guan, C.-C. Chiu, and S. Fu, "CDA: A cloud dependability analysis framework for characterizing system dependability in cloud computing infrastructures," in *Proc. IEEE 18th Pacific Rim International Symposium on Dependable Computing (PRDC'12)*, Nov. 2012, pp. 11–20.

[197] T. Hanawa, H. Koizumi, T. Banzai, M. Sato, S. Miura, T. Ishii, and H. Takamizawa, "Customizing virtual machine with fault injector by integrating with SpecC device model for a software testing environment D-Cloud," in *Proc. IEEE 16th Pacific Rim International Symposium on Dependable Computing (PRDC'10)*, Dec. 2010, pp. 47–54.

[198] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," in *Proc. International Conference on Dependable Systems and Networks, (DSN'06)*, 2006, pp. 249–258.

[199] R. Ghosh, F. Longo, V. Naik, and K. Trivedi, "Quantifying resiliency of IaaS cloud," in *Proc. 29th IEEE Symposium on Reliable Distributed Systems (SRDS'10)*, Oct. 2010, pp. 343–347.

[200] X. Kong, J. Huang, C. Lin, and P. Ungsunan, "Performance, fault-tolerance and scalability analysis of virtual infrastructure management system," in *Proc. 7th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA'09)*, May 2009, pp. 282–289.

[201] X. Kong, J. Huang, and C. Lin, "Comprehensive analysis of performance, fault-tolerance and scalability in grid resource management system," in *Proc. 8th International Conference on Grid and Cooperative Computing, (GCC'09)*, August 2009, pp. 83–90.

[202] S. Kaplan and B. J. Garrick, "On the quantitative definition of risk," *Risk Analysis*, vol. 1, no. 1, pp. 11–27, 1981.

[203] B. C. Ezell, S. P. Bennett, D. Von Winterfeldt, J. Sokolowski, and A. J. Collins, "Probabilistic risk analysis and terrorism risk," *Risk Analysis*, vol. 30, no. 4, pp. 575–589, 2010.

[204] P. Cholda, P. Guzik, and K. Rusek, "Risk mitigation in resilient networks," in *2014 6th International Workshop on Reliable Networks Design and Modeling (RNDM)*, Nov 2014, pp. 23–30.

[205] P. Choda and P. Jaglarz, "Energy-efficiency versus resilience: risk awareness view on dimensioning of optical networks with a sleep mode," *Photonic Network Communications*, vol. 30, no. 1, pp. 43–58, 2015.

[206] U. Onwudebelu and B. Chukuka, "Will adoption of cloud computing put the enterprise at risk?" in *Proc. IEEE 4th International Conference on Adaptive Science Technology (ICAST'12)*, Oct. 2012, pp. 82–85.

[207] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communicactions of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[208] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language," in *Proc. 8th USENIX Conference on Operating Systems Design and Implementation (OSDI'080)*, 2008.

[209] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*, 2010, pp. 135–146.

[210] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, and V. Sarkar, "X10: An object-oriented approach to non-uniform cluster computing," *ACM SIGPLAN Notices*, vol. 40, no. 10, pp. 519–538, Oct. 2005.

[211] T. Baker, M. Mackay, M. Randles, and A. Taleb-Bendiab, "Intention-oriented programming support for runtime adaptive autonomic cloud-based applications," *Elsevier, Computers and Electrical Engineering*, vol. 39, no. 7, pp. 2400 – 2412, Oct. 2013.

[212] H. Chai and W. Zhao, "Interaction patterns for Byzantine fault tolerance computing," in *Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition*, ser. Springer, Communications in Computer and Information Science, 2012, vol. 342, pp. 180–188.

[213] D. Cunningham, D. Grove, B. Herta, A. Iyengar, K. Kawachiya, H. Murata, V. Saraswat, M. Takeuchi, and O. Tardieu, "Resilient X10: Efficient failure-aware programming," in *Proc. 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'14)*, Feb. 2014, pp. 67–80.

[214] P. Narasimhan, K. Kihlstrom, L. Moser, and P. Melliar-Smith, "Providing support for survivable CORBA applications with the immune system," in *Proc. 19th IEEE International Conference on Distributed Computing Systems,(CSDL'99)*, June 1999, pp. 507–516.

[215] T. Akidau, A. Balikov, K. Bekiroğlu, S. Chernyak, J. Haberman, R. Lax, S. McVeety, D. Mills, P. Nordstrom, and S. Whittle, "MillWheel: Fault-tolerant stream processing at internet scale," *Proc. VLDB Endowment*, vol. 6, no. 11, pp. 1033–1044, August 2013.

[216] A. C. Squicciarini, G. Petracca, and E. Bertino, "Adaptive data protection in distributed systems," in *Proc. 3st ACM conference on Data and Application Security and Privacy (CODASPY'13)*, Feb. 2013, pp. 365–376.

[217] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*, April 2012, pp. 2–2.

[218] H. Jin, X. Yang, X.-H. Sun, and I. Raicu, "ADAPT: availability-aware mapreduce data placement for non-dedicated distributed computing," in *Proc. IEEE 32nd International Conference on Distributed Computing Systems (ICDCS'12)*, June 2012, pp. 516–525.

[219] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, "An Untold Story of Redundant Clouds: Making your service deployment truly reliable," in *Proc. 9th ACM Workshop on Hot Topics in Dependable Systems (HotDep'13)*, Nov. 2013, pp. 3:1–3:6.

[220] M. Wazzan and A. Fayoumi, "Service availability evaluation for a protection model in hybrid cloud computing architecture," in *Proc. International Symposium on Telecommunication Technologies (ISTT'12)*, 2012, pp. 307–312.

[221] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A case for cloud storage diversity," in *Proc. 1st ACM Symposium on Cloud Computing (SoCC'10)*, June 2010, pp. 229–240.

[222] Y. Hu, H. C. H. Chen, P. P. C. Lee, and Y. Tang, "NCCloud: Applying network coding for the storage repair in a cloud-of-clouds," in *Proc. 10th USENIX Conference on File and Storage Technologies (FAST'12)*, Feb. 2012, pp. 21–21.

[223] N. Bonvin, T. G. Papaioannou, and K. Aberer, "Dynamic cost-efficient replication in data clouds," in *Proc. 1st ACM Workshop on Automated Control for Datacenters and Clouds (ACDC '09)*, June 2009, pp. 49–56.

[224] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "SPANStore: Cost-effective geo-replicated storage spanning multiple cloud services," in *Proc. 24th ACM Symposium on Operating Systems Principles (SOSP'13)*, Nov. 2013, pp. 292–308.

[225] M. Vrable, S. Savage, and G. M. Voelker, "BlueSky: A cloud-backed file system for the enterprise," in *Proc. 10th USENIX Conference on File and Storage Technologies (FAST'12)*, Feb. 2012.

[226] S. Mu, K. Chen, P. Gao, F. Ye, Y. Wu, and W. Zheng, "uLibCloud: Providing high available and uniform accessing to multiple cloud storages," in *Proc. ACM/IEEE 13th International Conference on Grid Computing (GRID'12)*, Washington, DC, USA.

[227] N. Grozev and R. Buyya, "Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey," *Software: Practice and Experience*, vol. 44, no. 3, pp. 369–390, 2012.

[228] J. Li and B. Li, "Erasure coding for cloud storage systems: A survey," *Tsinghua Science and Technology*, vol. 18, no. 3, pp. 259–272, June 2013.

[229] C. Papagianni, A. Leivadeas, and S. Papavassiliou, "A cloud-oriented content delivery network paradigm: Modeling and assessment," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 287–300, 2013.

[230] R. Guerraoui and M. Yabandeh, "Independent faults in the cloud," in *Proc. 4th International Workshop on Large Scale Distributed Systems and Middleware*, ser. LADIS '10. ACM, 2010, pp. 12–17.

[231] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and secure storage in a cloud-of-clouds," *ACM Transactions on Storage*, vol. 9, no. 4, pp. 12:1–12:33, Nov. 2013.

[232] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys and Tutorials*, vol. PP, no. 99, pp. 1–1, 2015.

[233] C. Colman-Meixner, M. Tornatore, and B. Mukherjee, "Cloud-network disaster recovery against cascading failures," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'15)*, Dec. 2015.

[234] K. Meerja, A. Shami, and A. Refaey, "Hailing cloud empowered radio access networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 122–129, Feb. 2015.

[235] G. Koslovski, W.-L. Yeow, C. Westphal, T. Huu, J. Montagnat, and P. Vicat-Blanc, "Reliability support in virtual infrastructures," in *Proc. IEEE Second International Conference on Cloud Computing Technology and Science (CloudComm'10)*, 2010, pp. 49–58.

[236] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conference on Computer and Communications Security (CCS '07)*, Nov. 2007, pp. 584–597.