IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 4, 2019, accepted January 10, 2020, date of publication February 3, 2020, date of current version February 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971007

Architectural Resilience in Cloud, Fog and Edge Systems: A Survey

VICTOR PROKHORENKO[1] **AND M. ALI BABAR**[1,2]
[1]Centre for Research on Engineering Software Technologies, The University of Adelaide, Adelaide, SA 5005, Australia
[2]Cyber Security Research Centre Ltd, Kingston, ACT 2604, Australia

Corresponding author: Victor Prokhorenko (victor.prokhorenko@adelaide.edu.au)

This work was supported in part by the DST Group and Data61, which is Business Unit of CSIRO Australia. The work of M. Ali Babar was supported in part by the Cyber Security Research Centre Limited through the Australian Government's Cooperative Research Centers Program.

ABSTRACT An increasing number of large-scale distributed systems are being built by incorporating Cloud, Fog, and Edge computing. There is an important need of understanding how to ensure the resilience of systems built using Cloud, Fog, and Edge computing. This survey reports the state-of-the-art of architectural approaches that have been reported for ensuring the resilience of Cloud-, Fog- and Edge-based systems. This work reports a flexible taxonomy for reviewing architectural resilience approaches for distributed systems. In addition, this work also presents a capability-based cyber-foraging framework intended to improve the overall system resilience in the context of a physical node's capabilities. This survey also highlights the trust-related issues and solutions in the context of system resilience and reliability. This survey will help improve the understanding of the current state of system resilience solutions and raise awareness about the issues related to physical capabilities and trust management in the context of distributed systems resilience.

INDEX TERMS Cloud computing, cyber-foraging, cyber-physical systems, edge computing, system resilience.

I. INTRODUCTION

Systems based on traditional centralized cloud services architectures are becoming less efficient with the growth of service users and the amount of data required to be processed. Straightforward expansion of computing power behind cloud services cannot satisfy the growing performance expectations caused by the adoption of Internet of Things (IoT) devices and big data applications. Thus, alternative distributed computing approaches such as cloudlet-, edge-, fog-based and peer-to-peer computing are gaining popularity for building mission-critical systems.

Such systems are expected to be resilient to maintain an acceptable level of operations in hostile and disadvantaged environments. Designing and analyzing resilient software architectures of mission-critical systems are emerging areas of research and practice. Hence, there is a vital need of gaining an advanced understanding of various resilience characteristics of architectures for cloudlet, edge- and fog-based mission-critical systems. Such understanding is expected to develop and evaluate innovative and unique methods, approaches, and tools for designing and evaluating resilient architectures of such systems. In addition, a formal resilience definition and taxonomy is provided in Section III.

Based on a literature review, we have identified several resilient architectural approaches that can be classified into three main categories; capacity, trustworthiness and efficiency. Capacity-oriented solutions are primarily targeted to aid service providers by maximizing the number of users who can be serviced and minimizing the associated maintenance expenses. Virtualization techniques such as on-the-fly VM provisioning are a distinct feature of capacity-oriented approaches. The ease-of-use and affordability of modern third-party provided services makes it tempting to rely on them for multitude of tasks. However, military and other mission-critical scenarios may not accept the risks associated with sharing classified and private data.

Trustworthy architectures mainly focus on secure node communication and trust-related issues. The efficiency-oriented architectures attempt to improve the overall system operation by overcoming physical limitations inherent in a system's individual nodes. Furthermore, resilience is typically achieved at a higher level by a fault-tolerant node management such as node failure recovery techniques.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiping Hu.

Various forms of data and computation offloading are generally employed by efficiency-based approaches.

A. MAIN CONTRIBUTIONS

This survey has identified a large number of architectural resilience challenges and solutions for distributed systems. This work also determines the potentially beneficial areas for future research. The identified areas include developing a resilience-oriented architecture framework based on network node capabilities rather than on capacity, providing a formal definition for resilience metrics and improving resilience orchestration techniques.

The proposed capability-based architecture naturally extends to handle reverse-foraging scenarios such as when an edge node or a cloudlet may request video feed from end devices to locate some objects. Developing reliable resilience metrics is useful in comparing different architectures from the resilience perspective. Having an efficient run time resilience orchestration implemented allows to improve the overall system resilience and provide a higher level of system service operation.

The rest of this paper is organized as follows. An overview of various cloud system architectures is presented in Section II. General system resilience along with domain-specific resilience considerations are discussed in Sections III and IV respectively. Capability-aware resilience orchestration is examined in Section V. Section VI reviews the current research efforts with Section VII outlining the present challenges and opportunities. Lastly, Section VIII concludes this survey.

II. CLOUD SYSTEM ARCHITECTURES OVERVIEW

The number of inter-connected systems and devices through Intranet or Internet has been growing exponentially that has led us to a hyper-connected World. The hyper-connectivity of the current and increasingly future software systems poses new challenges for addressing the storage and processing needs, mostly in real-time. To address the increasing demands of the storage and processing needs, there are various technological options available such as cloud-based computing, cloudlets, edge computing and fog computing. These technologies are being leveraged for building and leveraging computing, storage and networking infrastructures in various ways to build large-scale distributed systems.

One of the first problems arising from the growing number of network nodes was the efficient data delivery. Early solutions to this problem involve deploying caching proxies within network boundaries of an organization. This can be seen as an early attempt to introducing an intermediate edge layer to optimize system performance. The rationale behind such an approach assumes that a significant amount of the same incoming data is requested by multiple network members. However, due to the diversity of modern web-based systems, the interest in generic web caching proxies significantly decreased. Whilst still used for optimizing data delivery and distribution, such caching proxies are now mainly used in specialized cases (e. g., antivirus or operating system update servers). In contrast to locally deployed proxies, service providers commonly use Content Delivery Networks (CDN) to optimize the data delivery to end-users.

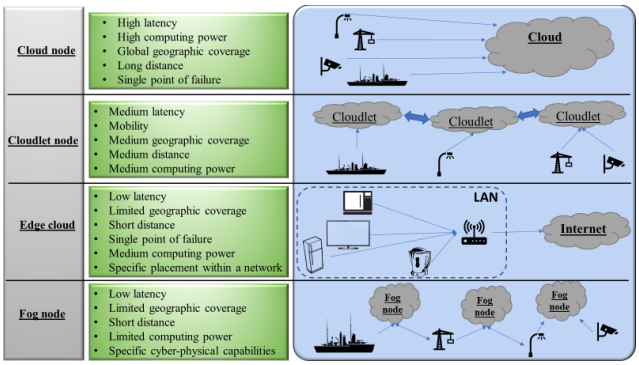

FIGURE 1. Key characteristics for cloud-, cloudlet-, edge- and fog-based system architectures.

Subsequently, the opposite problem of collecting user-generated content gained attention. Online streaming services are a notable example of services, which have to solve the problem of massive user-generated data collection (and possibly further redistribution). Data collection is complicated due to large number of users and massive amount of traffic required for high-definition video streams. These high requirements caused rapid advancement in traditional and mobile cloud technology, with a wide variety of types of service architectures being currently offered on the market and used in practice [1]–[4].

Lastly, in addition to the data collection and distribution problems, various forms of distributed computing also aim to solve such challenge as a lack of local resources. Typical examples of such resources include computational power and data storage. Generic resource-offloading such as storing a collection of photos taken by modern smartphones remotely, may provide satisfactory results in consumer applications. However, additional requirements arise for high-demanding real-time and military applications. Notably, in efforts to minimize the introduced delays cloud computing further evolved into fog computing [5], [6] and edge computing [7]. Whilst being somewhat different and applicable in different scenarios [8], these types of architectures share common requirements such as resilience, availability and efficiency. Figure 1 highlights typical use cases and key characteristics of cloud-, cloudlet-, edge- and fog-based architectures.

Cloud computing paradigm provides a convenient and a cost-effective way for low-powered devices to get access to nearly unlimited resources such as additional disk storage or processing power. From the end user perspective, however, cloud-based systems have two significant disadvantages; they can increase network latency and can become ''single point of failure''. In addition, centralized clouds may also be prone to cascading failures that cause higher stress on the remaining operational nodes [133], [134].

Cloudlet-based systems attempt to solve performance-related issues by bringing powerful computing nodes closer to computing consumers. The proximity of such nodes typically provides more control and enables cloudlet mobility, which is particularly useful in field environments. Deploying multiple cloudlets can also be used to overcome a single point of failure issue inherent in cloud systems.

Edge computing in general can be seen as a compromised approach, which also takes node ownership (and thus privacy aspects) into consideration. Similar to cloudlets, edge nodes are placed in close proximity of end nodes. Being located at the boundary of a local network, edge nodes are able to aggregate data from end nodes and perform some initial processing locally, while only accessing the external resources when necessary. This architecture suits smart homes and smart metering environments, with no specific node mobility requirements, but high privacy expectations.

A fog-based system architecture can be interpreted as an extension of the cloudlet approach with further focus on improving the overall system resilience. All the network nodes are kept in close proximity, with more granular task offloading from end nodes to fog nodes. In the extreme case, a fog-based network can be treated as a P2P network with specialized nodes. Naturally, depending on the type of tasks to be solved by a system and the operating environment various hybrid approaches may be employed.

Most of military applications are usually designed to operate in hostile environments, which can be defined based on certain contextual factors. Resilience of a system is especially important for systems used in hostile, disadvantaged and corporate environments [9], [11], [12]. Military systems' usage scenarios typically assume a worse connectivity for network nodes or a higher rate of node failures. Another distinct aspect of cyber-physical systems is the importance of node capabilities and physical placement. For example, a car and an airplane differ significantly in their physical capabilities. Sustaining an acceptable level of operation for such systems requires constant node capabilities monitoring and run-time system adjustments to compensate any failures.

This survey is aimed at critically reviewing the relevant literature on incorporating resilience in designing and implementing software intensive systems with/for cloud-based technologies such as cloudlets, fog computing, edge computing and peer-to-peer systems.

In contrast to taxonomies focused on system functionality, this survey attempts to expose the driving motivations behind architectural choices. The following diagram (Figure 2) illustrates the spectrum of various system architectures based on node proximity and availability.

According to this spectrum, a central cloud is an approach to consolidating all available resources in a central location without paying attention to the distance to user nodes. On the other end of the spectrum, peer-to-peer networks imply that no additional investments are required in individual nodes, with the main idea of placing nodes close to each other. The following sections discuss the difference between and applicability of various network architectures from the perspective of overall system resilience.

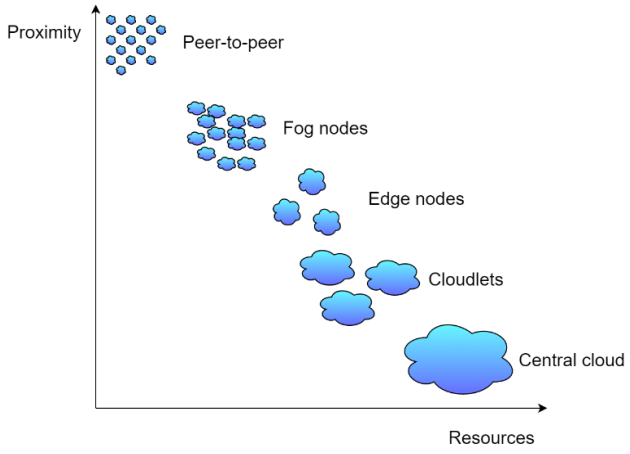

FIGURE 2. Various system architectures spectrum.

A. CLOUD COMPUTING

Centralized cloud-based solutions imply having a powerful central node, which can aid end-users with some specific functionality. Whilst from user perspective such a cloud may be visible as a single entity, cloud services providers may operate a complicated architecture internally, for example, with the use of load-balancing techniques.

A well-established terminology is used to refer to different types of cloud services. Common classification includes a multi-layered stack of Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).

Higher- or lower-level layers such as Business-as-a-Service or Management-as-a-Service are also commonly used to refer to more specific cloud services [12]. Commercial cloud solutions aim to lift off the burden of local resource management from consumers; therefore, substantial efforts are put into cloud services cost optimization [13].

Notably, the pay-per-use model becomes increasingly popular, to the extent of offering a per-second charging options available. Numerous optimization approaches are used by cloud service providers to reduce the associated cost. For example, using overprovisioning, virtualization techniques and dynamic resource provisioning allows servicing more customers with the same hardware. In addition to traditional encryption schemes, virtualization improves the overall system security by offering strong isolation between multiple clients which might not necessarily trust each other [14].

The largest downside of cloud computing is the possible lack of trust between cloud provider and consumers [15]. Privacy concerns can render the use of cloud services unsuitable in some scenarios depending on customer requirements or state legislation. Another concern some customers might have is vendor lock-in issue and a possible data loss. Whilst not specifically unique to cloud-based systems, the

importance of data processed and large number of customers served requires to pay high attention to the service availability and resilience [16], [17], [130], [136], [141], [143], [144].

Whilst cloud-based systems are commonly used to enable cyber-foraging (in form of data staging or computation offload) for weak user devices such as mobile phones, the advances in technology continuously improve the computational power of these devices. The network bandwidth and latency issues associated with cloud usage may lead to local computations being more efficient in certain cases. Therefore, cloudlet-based approach was introduced in order to minimize network-related overhead.

B. CLOUDLETS

Cloudlet-based approach originates from the efforts to reduce the latency issue inherent in centralized clouds [18]. Bringing cloudlet nodes closer to end nodes can lead to significant processing improvements [19], [20]. Such optimization, however, comes at a cost of more cloudlet nodes to be employed. Cloudlet instances are typically less powerful than a centralized cloud and in compared to clouds, cloudlets also serve a lower number of nodes. Operating multi-layered architectures of different cloudlets being connected to a centralized cloud requires introducing fault tolerance to improve the overall system resilience in hostile environments [10], [21].

Optimal cloudlet placement is essential for cloudlet performance and is usually performed at two layers, physical and logical. Taking the physical layer into consideration is important for networks consisting of mobile nodes such as for tactical missions [22]. The logical layer cloudlet placement does not directly rely on physical node locations and is instead focused on dynamic cloudlet provisioning based on current demands [3].

Whilst the logical-level placement is well-researched and highly automated, physical-level placement decision algorithms tend to overlook the physical characteristics (including location and moving speed) of network nodes. Deeper research in this area would be potentially beneficial to cyber-physical systems, including military infrastructure, smart cities, smart health and transportation.

From the end device perspective, cloudlet node discovery issue arises due to the dynamic nature of cloudlet location. End nodes must be able to dynamically and reliably find the closest trusted cloudlet. Various protocol-level implementations such as Bonjour and Avahi, were introduced to achieve node and service discovery. However, depending on cloudlet ownership, lack of trust towards a cloudlet may become an obstacle, leading to self-hosted cloudlets being more suitable compared to third-party provided ones. Further optimization steps taken by Edge and Fog computing are discussed in the next section.

C. EDGE AND FOG COMPUTING

Compared to cloudlet-based systems, both Fog and Edge computing take a further step toward bringing processing nodes even closer to end nodes. An obvious advantage of such optimization lies in the reduced traffic usage and latency. Both Edge and Fog approaches aim to complete as much data processing as possible in the immediate proximity of end nodes. Similar to cloudlets, fog nodes can be connected in a multi-tier hierarchy, with further data processing being offloaded towards more powerful cloud nodes when necessary.

At the technical level, Fog and Edge paradigms are quite similar with some works even referring to them interchangeably [23]. However, certain terminology-related disagreements exist with regards to definitions and difference between Fog and Edge computing [24]–[26]. Moreover, in some works, a resilience of a system is impractically defined (partially) through the ability of a system to resist various failures [27]. One of the views is that Fog computing can be perceived as an extension of cloud computing [28].

The main difference between Fog and Edge approaches is the emphasis of the special node placement. More specifically, Edge nodes are supposed to be placed at the edge of a network. In scenarios such as smart house management, Edge nodes can be placed within the boundaries of end-user's network. For mobile devices, however, Edge nodes have to be placed within the boundaries of service providers [29]. This makes Edge computing particularly suitable in areas similar to cellular networks and smart transportation [30]–[32], [147]. Fog paradigm does not imply specific node placement constraints or expectations, instead Fog nodes can be located anywhere in close proximity of end nodes, which makes Fog computing useful for IoT based applications.

Due to being located outside of well-protected service providers network, Fog nodes may encounter additional security and trust issues [33]. Likewise, reliability and fault tolerance related challenges and solutions for Fog nodes are currently heavily researched [34], [35].

In summary, the choice of a system architecture is initially dictated by its tasks with resilience enhancing techniques applied at a later stage. Currently, global system adjustments are performed manually as user demands evolve over time. For example, the Bitcoin network was initially designed having peer-to-peer architecture. Subsequently, with the growth of the number of network users and transactions, the overall system performance became insufficient. One of the proposed generic architectural solutions to overcome the performance limitations involves introducing an intermediate layer of more powerful super-nodes. Possessing higher computational or storage resources, these super-nodes can essentially be perceived as cloudlets. Given the constantly changing user demands, automated and mission-aware system reconfiguration at run time should be needed.

Besides affecting the overall operating performance, the selected system architecture can also indirectly affect a number of other factors such as reliability, availability or resilience. The following section focuses on the resilience

factors affected by the choice of the discussed system architectures.

III. SYSTEM RESILIENCE

A system's resilience is characterized and assessed based on the main purpose that system serves. For example, backup solutions are intended to be resilient against data loss, but are not necessarily useful against data leakage. Whilst some literature uses the terms resilience and reliability interchangeably, in the context of cloud solutions the reliability is commonly treated as one of the desired system features. Resilience is the process required to achieve the desired outcome in the form of reliability.

Complex distributed systems deal with a variety of issues which may affect normal system operation; that means resilient system implementations must address multiple factors. Depending on the type of services to be provided by a system such factors may include, but are not limited to weak (or interrupted) connectivity, limited power, and Advanced Persistent Threats (APT). In practice, resilience solutions usually focus on a subset of disruptive factors.

Traditionally two types of faults are considered. Namely, crashes and byzantine faults with similar solutions used against both types. Due to the high costs and latency Byzantine Fault Tolerance (BFT) is not currently used within data centers. Instead, specific fault-oriented solutions are typically employed in practice. Distributed cloud-based solutions may encounter faults of both types at all levels, such as storage, network, malicious actors, and power faults. Based on a large-scale server fault survey [36] focuses on node placement within a data center clusters and relies on virtualization techniques in order to achieve recoverability.

Based on this view, the general fault-tolerance mechanism architecture is supposed to consist of two layers. Namely the business logic layer which is responsible for normal system operation and meta-layer, which is responsible for fault handling and recovery [37], [63], [123]. This allows more flexible and continuous deployment of fault tolerance to existing or developed systems. A commonly overlooked factor is the resilience of the monitoring agents/modules themselves.

Further fault sub-classification could depend on factors such as longevity of a fault, which distinguishes between transient and permanent faults. Some approaches focus on so-called grey failures instead of fault-stop failures [38]. Rather than focusing on resilience against certain types of service disruptions, a purpose-oriented resilience classification is proposed.

Based on the types of disruptive factors addressed, software intensive resilience solutions can be categories in three large classes as shown in Figure 3 - Capacity-, Trustworthiness- and Efficiency-oriented.

- Capacity – this means resilience against legitimate system uses. Here, the ''capacity'' refers to the large amount of data being processed and large number of users served.

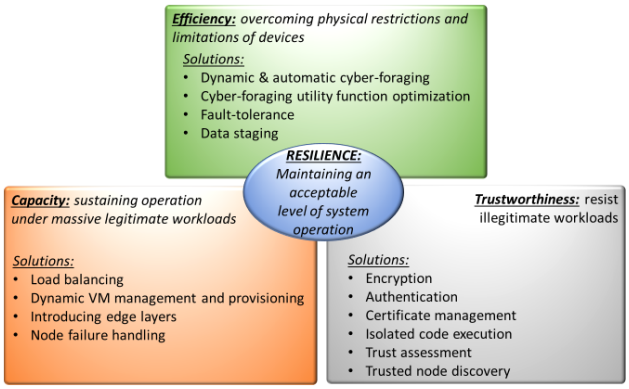

FIGURE 3. Types of resilience.

- Trustworthiness – this means resilience against unwanted system uses. Typical resilience measures of this class include encryption, certificates and trust-oriented approaches.
- Efficiency – this means resilience against physical restrictions and limitations. Typical examples of such restrictions are unstable connection links (radio), low-capacity batteries and natural disasters.

The efficiency of service operation can be considered one of the main driving forces behind the popularity of distributed computing in general. Efficiency-oriented approaches aim to overcome various physical limitations. Typical examples of such limitations are possible natural disasters, weak connectivity links, device battery capacity, computing power, disk storage and timing constraints. In cases, when an acceptable service operation level cannot be achieved using the locally available resources, cyber-foraging approaches are used. Cyber-foraging involves using remote node computational or other resources in situations when local resources are not adequate for a task. The cyber-foraging can be either implemented as part of an architecture at system design stage or introduced to existing legacy systems with specialized tools [39].

Capacity-oriented solutions attempt to solve the problems related to the growing number of network nodes, increasing traffic demands and high computing power requirements. Modern high-quality image, video and audio processing services typically face such issues. The terms ''elasticity'' and ''scalability'' are commonly used to refer to such relevant solutions. Conceptually, capacity-oriented solutions focus on increasing a service provider-side resilience, to increase a service's ability to maintain acceptable level of operation under high load. Such types of solutions try to minimize a service's maintenance and management costs.

Lastly, the trustworthiness-related challenges and risks have also gained significant attention [29]. In addition to protecting the individual nodes' operations, the inter-node communication must also be protected. Loss of control over the offloaded or shared data also raises trust- and privacy-related problems [14], [33], [40]. Distinct features

of trustworthiness-oriented approaches include the use of encryption, cryptographic signatures and certificates. The following sections provide a detailed review of the literature on each of the categories.

A. TRUSTWORTHINESS

Trust in general is a subjective measure of probability of a particular node to perform certain operations [41]. A common approach is to consider trust directional and not necessarily transitive. Furthermore, dynamic and distributed systems pose additional challenges as the list and locations of network nodes is not known in advance. Overall, trust-related solutions can be broadly classified into two categories - trust-ensuring procedures and trust-less operation.

Trust-ensuring procedures aim to abstract trust away to the underlying layers and maintain further system operation based on the established trust relations. Various dynamic rating systems such as trust and reputation management are typically used to establish such relations [42]. Whilst being useful in distributed networks with all the nodes belonging to the same party, this type of approaches may not be suitable for multi-party networks. Instead, a pseudo-trust can be established and used in such scenarios [43]. The main disadvantage of trust-establishing solutions is the requirement to either rely on a pre-shared knowledge about nodes (certificates) or collect reputation statistics at run-time. An alternative trust-maintaining approach based on the moving target concept was proposed in [44].

Trust-less system operation approaches tend to keep system operation without relying on the honesty of remote nodes. Rather than blindly trusting remote nodes, such approaches attempt to verify the results of remote node computations [45], [46]. The main disadvantage of computation verification is the increased financial or computational overhead [47]. Depending on the type of the computations involved, the technical implementation of the verification procedure may become impractical [48].

For simple types of services involving only a data offloading component, the trust aspect is typically limited to authentication, encrypting, cryptographically hashing and signing the offloaded data [49]–[52]. Complicated data processing becomes challenging when untrusted surrogates are considered [53], [54]. Privacy cautious users may refuse to decrypt sensitive data on a surrogate node. A promising solution for some types of data processing is homomorphic encryption [55]. Unfortunately, the currently researched homomorphic encryption techniques are computationally-intensive. Therefore, homomorphic encryption is currently not well-suited for low-powered mobile devices, which are quite pervasive in an infrastructure built with Edge or Fog computing.

Additional complications arise when the type of data processing required is not known and available for the surrogate nodes in advance. Generic code execution is commonly used in such scenarios to achieve greater level of service flexibility. In addition to the data itself, the end-devices are expected to provide the data-processing code to surrogate nodes. To minimize a possible negative impact, surrogate nodes need to isolate the received code (typically using virtualization techniques) prior to execution [56].

The trust issues are twofold: on the one hand, the end devices may not fully trust surrogate nodes. On the other hand, surrogate nodes may not trust the code supplied by end devices. Authentication, code signing and social-based techniques can be used to address lack of trust [57], [58]. These approaches imply that some form of registration is required for end-user devices to be able to use surrogate nodes. Techniques addressing this type of lack of trust have not been extensively investigated.

For example, encrypting, hashing or cryptographically signing the offloaded data may be adequate to protect data from untrusted surrogate nodes. However, a possible loss of the offloaded data becomes a threat. Unlike typical commercial solutions that use data backup systems behind the scene, a malicious surrogate could intentionally delete the uploaded data. Similarly, such a malicious surrogate may also modify the data processing routines and return invalid or fake results. Therefore, formal computation result verification procedures might be required to allow the use of untrusted surrogate nodes.

In contrast to publicly accessible services, using self-owned systems (where both the end and edge devices belong to the same party) allows to minimize trust issues. Therefore, although possibly being more expensive to maintain than third-party owned services, self-owned systems may be more suitable for military environments. Whilst inter-surrogate communication is expected to be encrypted, some approaches do not explicitly state this requirement [9], [59].

B. CAPACITY

Network systems used in high-load environments must provide resilience against massive legitimate system workloads (such as DoS and DDoS attacks). Therefore, some approaches focus on maximizing the number of users or amount of data that can be serviced simultaneously. System survivability is mainly achieved by implementing fault-tolerance and redundancy [60]. The techniques common to such approaches include dynamic Virtual Machine (VM) management, on-demand VM provisioning and load balancing [61]. In addition to minimizing the associated running costs [59], [62], these techniques make it possible to implement highly scalable services. By utilizing virtualization techniques, some solutions may improve the overall survivability of a system [20]. Rather than focusing on the service or application level, some approaches attempt to achieve resilience at network level [63]–[65].

In simple cases, an intermediate edge layer may be introduced to implement system service load-balancing [66]. More complex architectures, however, may utilize an additional layer (in form of cloudlets or special edge nodes) for computations [9]. Given the distributed and mobile nature of modern networks, surrogate node discovery itself becomes

crucial to allow optimal calculations or data offloading [67]. Specific surrogate node placement may also be used to simplify the discovery process.

Depending on the type of the services provided, surrogate nodes may either perform a predefined type of data processing or allow arbitrary code execution [9]. Such flexible data processing is commonly achieved by optimizing VM deployment and delivery. In addition to ease of use and low costs, VM-based approaches also provide code isolation.

Whilst providing more flexibility, arbitrary code execution on a remote surrogate node raises a number of security-related issues. However, the capacity-oriented solutions [68]–[70] for improving the overall system resilience usually do not take security into consideration. The approaches falling in the capacity category are focused on service providers' interests such as minimizing operational costs or increasing service coverage, and are well-suited for publicly accessible type of services.

System's QoS (Quality of Service) is somewhat similar to efficiency-oriented resilience, however, some key differences can be pointed out. Most notably, QoS-based approaches assume at least some level of system operation being available, rather than attempting to conduct recovery procedures. The main purpose of QoS policies is to prioritize various system functions with sacrificing less important ones if necessary. A comprehensive review of various resilience prioritization frameworks in the context of communication networks is presented in [11]. In this regard, QoS approaches are similar to graceful degradation approaches.

C. EFFICIENCY

The solutions falling into the category of efficiency address resilience from the perspective of the limitations of physical environment and end devices, i.e., mobile and sensor devices. Such solutions attempt to maintain an acceptable level of system operation despite changes in the physical environment. These changes may also be related to system nodes themselves. Typical examples include hardware failures and low battery levels. In other words, efficiency-oriented resilience approaches attempt to utilize the available resources in an *efficient manner* in order to achieve a system's goals.

Popular class of such solutions offload data or calculations to remote surrogate nodes due to certain limitations of end devices [21], [71]. Depending on the type of computation required to be offloaded, application, module or function-level offloading may be implemented [72], [73]. CPU processing power, disk storage and other resources are typically considered by cyber-foraging implementations [74]. However, with the battery capacity improvements significantly lagging behind the growth of CPU power, the end device battery level and consumption gains high attention [75]–[77]. Interestingly, efficiency-oriented approaches typically do not explicitly take the limitations of the surrogate nodes into consideration [78].

Due to the inherent network latencies and cyber-foraging overhead, computation and data offloading is typically performed only when it is estimated to be efficient [79], [80]. Note that recent advances in communication technologies may make computation offloading more practical and affordable. For instance, the high bandwidth provided by 5G technology is likely to lead to wider adoption of cyber-foraging. The priority of the assessed properties is based on a solution, system configuration or user preferences [81]. For example, a system's user may prefer to get computation results as fast as possible, regardless of the battery charge level. A special utility function is used to decide whether or not cyber-foraging is beneficial under certain conditions. Whilst the resources consumed by the utility function may be negligible, such resources consumption are rarely considered.

Cyber-foraging in general assumes that, albeit suboptimal, offloaded computations can be performed locally in case of surrogate failure. Therefore, resilience approaches commonly include fallback tactics to complete computations locally [81]. In contrast to traditional cloud-based systems (with nodes only differing in their capacity), complex systems used in military, rescue or industrial operations may consist of nodes that have different physical capabilities [82], [83]. Having nodes with different capabilities leads to some tasks being impossible to complete on certain nodes. For example, obtaining a video stream is only possible from the nodes that have a video camera attached to them.

Based on this distinction between node capacity and capability, cyber-foraging can be classified into two categories – optimistic and forced. Optimistic cyber-foraging essentially assumes that offloading may improve end-user experience, but is not strictly required for system functioning. Forced cyber-foraging can be viewed as an integral part of system operation. Therefore, advanced resilience tactics are required to tolerate forced cyber-foraging failures, as local fallback may not be possible. Such sophisticated resilience tactics include re-routing the requests further to other capable surrogates when possible [35].

An important component of these tactics is the surrogate node placement and discovery [10]. Apart from improving the network latency, the surrogate discovery can be simplified with an optimal node placement. Compared to classic Cloud-based systems, Cloudlet-, Edge- and Fog-based systems purport to improve the reachability of surrogates at the expense of introducing the node discovery process.

Lastly, a special case of reverse foraging (when a cloudlet initiates a request to one or more end nodes) can be reformulated in terms of system workflow reconfiguration. As a part of a universal distributed system architecture framework, system workflows are discussed in more detail in Section 5B. The next section focuses on context-specific resilience solution approaches.

IV. LAYER-SPECIFIC RESILIENCE

Whilst universal resilience metrics suitable for arbitrary systems have not yet been developed, various specialized metrics are commonly used in domain- and layer-specific scenarios [106]. Depending on the type of systems, resilience

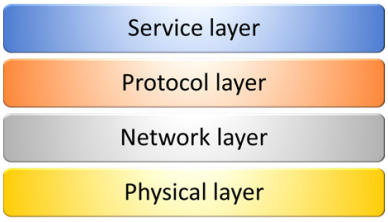

FIGURE 4. Resilience layers.

may be achieved at different layers against different types of potential issues. System resilience is achieved through implementing preventive and restorative measures against various faults. Thus, the term ''fault-tolerance'' is sometimes used to refer to resilience solutions. However, ''faults'' may have different nature at different layers of a distributed system and constitute only a subset of resilience threats. An overview of various resilience implementations layers identified through this survey is shown in Figure 4.

Faults at the physical layer include such events as power loss, hardware failures and natural disasters. Therefore, resilience solutions focused at physical layer mainly aim to address these types of faults. Common techniques used against physical faults are redundancy (including load-balancing) and backups. Whilst a significant portion of manual labour is still required to recover from severe physical failures, software automation is used to simplify some of the monitoring, reporting and restorative steps. Likewise, a set of preventative measures can be implemented in a system at design time in order to improve a system's overall resilience. Notable examples of such measures include SSD wear-levelling, RAM error correction approaches [84], RAID disk configurations and use of uninterruptible power supply units. Interestingly, similar preventative measures can also be taken against malicious actions rather than hardware failures [139].

Network layer faults disrupt communication between system nodes. Hence, node reachability is a resilience metric for network-level faults. Reachability fault can either be permanent or temporary, which can be expressed in terms of traffic-related degradation [60]. Various quantitative network metrics such as frequency of failures, average outage duration or packet loss rate can be used to determine a system's overall survivability at network level [85].

For depending on network topology, the probabilistic approaches such as churn rate can be used to measure a system's overall fault-tolerance [86], [87]. The main recovery technique used at the network level involves dynamically rerouting nodes communication [88], [89], [137]. Wireless networks pose additional challenges related to the nature of data transmission medium. Thus, wireless-specific protocol-level resiliency improvement techniques are being proposed for this type of scenarios [90].

Unlike network-level faults which affect the routing of node communication, protocol-level faults alter node communication contents [91]. Protocol-level resilience solutions assume that a communication channel exists, albeit unstable in terms of bandwidth or latency. Primary principles behind protocol-level resilience implementations are graceful degradation and error-tolerant data coding. Graceful degradation is typically used in audio/video transmission and involves reducing data bitrate to accommodate for lower bandwidth available. A promising technique which may become potentially achievable with Machine Learning (ML) would be to achieve bitrate reduction by switching to a different type of communication such as transmitting textual representation instead of video. Domain-specific lossless noise-resistant and error-correcting data coding techniques are used in cases when partial degradation is not acceptable. Whilst protocol-level resilience solutions mainly lie in the communication and signal processing domains, local caching/buffering techniques enable a service to operate in the absence of a connection can also be implemented to improve protocol-level resilience.

Lastly, service-level faults highly depend on the nature of service provided by a system. Implementing resilience for web-based applications may be significantly different compared to IPTV-system resilience. For instance, improving resilience against Denial-of-Service attacks for web applications can be implemented through graceful degradation by serving static pages as opposed to complex dynamic content. Naturally, this type of solution would not be reasonable for video streaming. A combination of peer-to-peer architecture and adaptive bitrate coding can be used instead in order to achieve resilience for video streaming systems. Having an in-depth knowledge of the provided services makes it possible to efficiently implement redundancy and graceful degradation [92]–[95].

Some resilience solutions focus on layer-agnostic issues such as human errors, performance or system security in general. Typical measures against human errors include pre-run verification and validation [96]. Cryptographic primitives such as encryption, signing and authentication are traditionally used to prevent unauthorized and malicious misuse of a system [97]. Dynamic resource usage optimizations such as cyber-foraging or on-demand VM provisioning are generally used to address a system's performance resilience.

A summary of the threats and their corresponding resilience-improving techniques is outlined in Table I. Despite domain-specific resilience monitoring and management being implemented and used in individual systems, a comprehensive and universal architecture-level solution is yet to be developed. Some resilience implementations aim to achieve overall system resilience by focusing on the resilience of the lower levels only [98].

Layer-based resilience encapsulation is quite common, with the Internet itself being a vivid example. Physical-level resilience is managed by individual node administrators, network-level resilience is achieved through multiple routes, data checksums are used to accomplish protocol-level resilience. Thus, in simple scenarios, individual services can rely on the existing resilience measures implemented at lower

TABLE 1. Resilience threats and solutions.

	Fault	Performance	Human error	Malicious misuse
Service	Graceful degradation	Cyber-foraging	Validation	Access control
Protocol	Error-correction data encoding	Graceful degradation (compression)	Verification	Encryption
Network	Rerouting	Redundancy (multiple links)	Redundancy	Authentication
Physical	Redundancy (manual recovery)	Redundancy (on-demand provisioning)	Redundancy	Physical access control

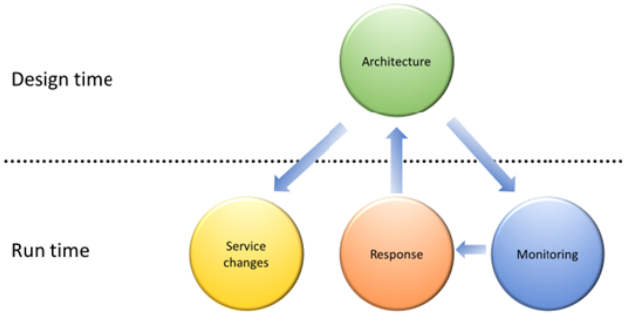

FIGURE 5. System resilience orchestration process.

levels. In contrast, complicated workflows inherent in highly distributed cyber-systems, may require a system to be aware of lower-level failures. A holistic universal resilience solution must coherently address failures occurring at every layer. An extensible resilience orchestration approach based on individual node capabilities and overall system goals is discussed in the following section.

V. RESILIENCE ORCHESTRATION

A system needs to be continuously monitored to detect possible system service level degradation for taking corrective actions. The monitoring and response procedures highly depend on a system's architecture and the types of provided services. Therefore, a flexible system architecture definition method should be developed in order to implement a universal orchestration approach. Figure 5 illustrates the proposed design- and run-time components of system resilience orchestration.

During the system design phase, a general system architecture is defined based on the available resources and the type of services to be provided. During a system's operations, the monitoring and corresponding restorative response procedures are continuously carried out. Externally imposed service changes such as new resources become available, may occur at any time during a system's operation. An efficient orchestration must be able to adapt to run-time changes and reconfigure a system as required. The following subsections discuss design-time architecture definition and run-time system adaptation phases.

A. SYSTEM ARCHITECTURE DEFINITION

Commercial applications normally tend to solve a specific user problem with cloud-based computing. The solution to each specific problem may require a different system architecture. Typical solutions provided for the scenarios involving modern smartphones are well-suited for user data acquisition and redistribution. Some works explicitly state that upon a successful data processing, the results need to be returned back to mobile devices [99].

However, mobile devices may not always require the results from the processed data. For example, the data acquired by drones is required to be sent to a ship not because the ship has a more powerful computer on board, but because the ship has some physical capabilities (missiles) which drones lack. Regardless of where the computations take place, physical capabilities strongly affect the flow of data in cyber-physical systems.

A universal framework designed to define the overall system architecture is currently not available. The two key components of such a framework are the formal definitions of the available resources and a formal task (mission) description. In some cases, accomplishing a given mission with the provided resources may not be possible. In those cases, a generic framework may also be useful to identify and recommend the minimal set of resources required to be added to the pool of available ones.

An example formal definition of a synthetic system mission and resources available is shown in Figure 6. In contrast to pure software systems, cyber-physical systems are likely to have a richer set of mission-critical capabilities (as opposed to commonly used set of CPU power, memory and network speed). Analyzing the available resources and a system's tasks can essentially produce a set of recommendations for physical investments into certain (possibly new) nodes. Broadly speaking cloud-, cloudlet-, edge- and fog-computing are generic attempts to invest into physical capabilities of certain nodes only. Instead of demanding each smartphone to be replaced with a faster CPU only a small number of powerful nodes are placed within a system. Such solutions aim to solve a system's tasks with minimal investment.

A preliminary static analysis can be conducted based on the formal definitions alone; whether or not the mission goals can be achieved with the available resources. This analysis can potentially lead to a set of system optimization recommendations such as alternative investment options or specific system node placement. Given that the nature of a mission and the available resources may change at arbitrary points in time, the dynamic system adaptation and reconfiguration

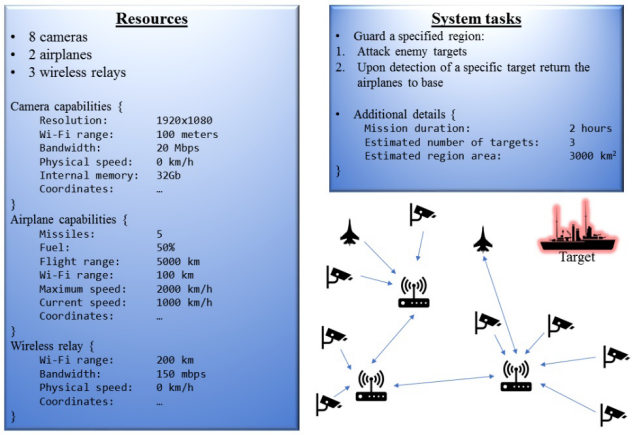

FIGURE 6. Format system resources and mission definition.

must also incorporate such analysis and provide updated recommendations at run time. The next section discusses the run-time system resilience orchestration framework aspects.

B. RUN-TIME SYSTEM ADAPTATION AND RECONFIGURATION

A system is expected to address failure recovery and dynamic adaptation issues for effective orchestration. Failure recovery mechanisms include graceful degradation techniques to keep a system at least partially operational while the recovery steps are being taken. Provided a system's operational level metrics are formally defined and continuously measured at run time, some of the steps can be taken proactively. For example, in cases when a critically high system load is detected, a system may choose to spawn additional VMs to improve the system's overall performance.

System heterogeneity may pose additional technical issues as low-level protocol implementations must be compatible across different nodes. These challenges may be further aggravated when multi-party systems are taken into consideration. Therefore, a set of universal agreements and protocols must be defined to allow effective interoperability. Theoretical system stability ensuring approaches have been proposed at abstract level [100]. Whilst some domain-specific attempts such as SNMP have been implemented, a comprehensive bidirectional (for data acquisition and control) capability-aware protocol implementation is yet to be developed.

It is relatively easier to implement a single control center system architecture, which can be a single point-of-failure. More specifically, in situations when the control center node becomes unavailable, the rest of the system's nodes may also become completely inoperable. Data caching and local fallback techniques are commonly used to counter potential failures of a central cloud. Such techniques, however, depend solely on node connectivity factor and may not always be suitable in highly distributed systems consisting of nodes with different capabilities.

System nodes capabilities define data- and work-flows in a system. Figure 7 highlights a typical workflow based on

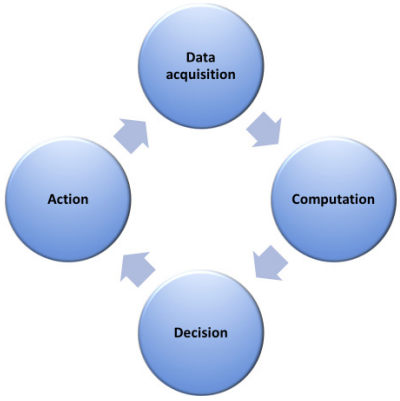

FIGURE 7. System workflow phases.

node capabilities and roles. Maintaining system resilience comprises of keeping each phase operational. Obviously not all events can be counteracted; for example, if a specific sensor node fails, there might be no spare nodes available that possess the same physical capabilities. However, recovery procedures such as electing a new node responsible for certain decisions, in case the original node fails can greatly improve a system's overall survivability.

Possessing a deeper understanding of the underlying system workflow is critical for cyber-physical systems where different nodes may considerably differ in their capabilities. For example, losing connectivity to a node with a powerful CPU may lead to slower processing times, whereas losing connectivity to a node with a video camera may render a surveillance system completely ineffective. Capability-based analysis of system operation is currently performed implicitly by system administrators and requires considerable amount of manual labor. Thus, there is a need of further research in the area of system orchestration automation. The next section overviews the current research efforts in design- and run-time components of system resilience orchestration.

VI. CURRENT RESEARCH EFFORTS

Some techniques, while not specifically targeting a system's resilience, may in fact increase the overall resilience as a side-effect. For instance, local caching and computation can be implemented in order to improve user experience (by minimizing network-related latency), however, this would also mean that a system may remain operational even if a cloudlet is not reachable by a client node.

In fact, various system improvements tend to be introduced to achieve resilience against some specific issues under certain conditions. A common example is introducing an encryption layer to an existing system. Whilst rarely positioned as a resilience measure in general, encryption obviously improves a system's global resilience by allowing a system to operate in presence of malicious nodes. The classification of system resilience improvement research directions discussed in this section is presented in Figure 8.

System resilience improvement approaches can be categorized into design-time and run-time. Design-time approaches

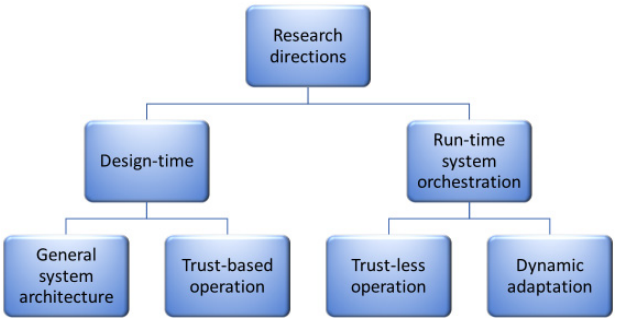

FIGURE 8. System workflow phases.

focus on the initial system configuration which include general architecture (such as a single centralized cloud vs. multiple cloudlets) and embedded trust (such as a central authority). Run-time system orchestration approaches tend to rely less on the initial system configuration and are instead operating under an assumption that the system configuration may change drastically at any time.

A. DESIGN-TIME

Design-time decisions define the general system operation and aim to solve major problems of node placement and node collaboration. Given some existing set of resources, an optimal node placement must be selected. This placement depends on node capabilities/roles and varies from fully-centralized to fully-distributed (Figure 2). However, users' demands and system requirements may change significantly during system operation.

Unfortunately, there is not much attention being paid to formalizing and automating architectural decisions. Automating the node placement and resource redistribution task requires a profound knowledge of a system mission. A higher-level understanding of service availability currently used by software availability monitoring tools is not adequate for cyber-physical systems containing nodes with vastly different capabilities.

The second problem required to be solved at design time is efficient node collaboration. In addition to the technical definition of low-level network protocols, a trusted relationship must be established between system nodes. Some research efforts are focused on implementing trust as part of a system's architecture. An opposite approach of implementing a system's operation independent of initial trust at run time is discussed in Section 6B.

It has already been stated that a central authority-based trust management is prone to a single point-of-failure problem. Thus, resilient solutions need to maintain trust in a distributed manner. The key properties of trust are subjectivity, observability and dynamic nature [101]. Whilst trust is generally not considered transitive, it is possible to introduce conditional transitiveness for certain scenarios [41]. A common feature inherent to design-time trust-based approaches is some prior knowledge embedded in a system such as root certificates or a list of bootstrap servers. Distributing this prior knowledge between system nodes becomes challenging for highly dynamic systems, where a node can join or leave a system at any time. Hence, various run-time trust management techniques have been proposed.

B. RUN-TIME

The two key problems that need to be addressed during a system's operation are trust establishing and adapting to environment or mission changes. Establishing at least some form of trust between system nodes is required when no initial trust such as a central authority, is embedded in a system architecture. Moreover, due to a possibly dynamic nature of a system, it becomes crucial to continuously maintain the established trust during a system's operation.

1) TRUST-LESS OPERATION

Before identification, authentication and authorization procedures can be carried out between system nodes, the nodes must be able to locate each other. This step is known as node or service discovery and various protocol-level solutions have been developed. Extending these discovery mechanisms to take node physical capabilities into account would significantly improve their applicability to cyber-physical systems. Once a participating node has been discovered, the trust aspect must be addressed. There are two general approaches to managing trust-related issues in systems with no inherent trust embedded. First is to actually establish some form of trust at run tie and second is to operate in a trust-less manner.

A common technique to introduce pseudo-trust to a system is to assign a reputation to participating nodes, based on their previous activity and recommendations of other nodes. Some works explicitly do not distinguish between trust and reputation management [42]. Trust-establishing mechanisms typically assume that network layer guarantees data delivery and attempt to address following issues; asymmetric nature of trust, selfish nodes refusing to participate and collusion of malicious nodes [43].

In contrast to establishing trust at run-time, trust-less approaches attempt to take advantage of even completely untrusted nodes. This is achieved either through homomorphic encryption or computation verification. Using homomorphic encryption, however, does not prove that computation itself was performed properly as a malicious node may return fake or incomplete results. Increasing resilience against such malicious and unreliable nodes is achieved through computational verification.

Assuming that malicious nodes operate independently, a simple majority voting can be implemented by cross-checking the results originating from different nodes [47]. An obvious downside of such approach may be an increased cost of operation for pay-per-use services. Naturally, the result checking procedure must be significantly cheaper than performing computation itself [45], [46]. Despite the high computational cost (and consequently increased power requirements) associated with result verification, there are promising recent advances in this area [48].

Perhaps the largest disadvantage of trust-less approaches is high cost of error. More research in this area is required to improve a system's resilience in hostile environments.

The discussed solutions address the issues of trust of end nodes towards surrogate nodes. However, trust issue is two-fold. On one hand, end-nodes may not trust data or result integrity. On the other hand, surrogate nodes performing the computation may not trust the code provided by end nodes. Thus, various code isolation techniques such as virtualization are commonly applied to surrogate nodes.

2) DYNAMIC ADAPTATION

There may be significant changes in a system's environment that necessitate adaptation to the changes, which can be categorized into scale-oriented and mission-oriented. Scale-oriented changes include node and resource variations. For instance, if some surrogate nodes fail, the overall computational power of a system decreases. The key principles behind handling scale-oriented changes are fault tolerance achieved through redundancy and graceful degradation. Taking advantage of new nodes and resources makes it possible to optimize a system's performance [102]–[104]. If a node and its associated resource fail, the required resources can be acquired from newly added nodes using load-balancing.

Mission changes originate from a system's owners and reflect new goals. These changes may occur independent of a system's scale, thus requiring some resource redistribution to be conducted. Currently, such resource redistribution techniques largely overlook physical node capabilities and are mostly focused on dynamic VM provisioning. However, abstracting physical node capabilities through a virtualization layer, restricts the applicability of such solutions to purely software domain.

Table 2 provides a classification of the reviewed resilience solutions based on the implemented resilience's type and architectural level. Note that due to being out of software domain, physical-level resilience solutions such as electronic locks, security guards, physical resource protection and video surveillance are not considered here. As can be seen in Table 2, the efficiency- and capacity-oriented approaches are more popular than the trustworthiness-oriented solutions. Similarly, service-level solutions are more popular than lower-level protocol- and network-based ones.

Whilst hybrid solutions covering different resilience aspects and layers are somewhat common, none of the reviewed work addresses all of the resilience types and layers. Thus, additional research is required to develop a comprehensive and efficient resilience solution. Another potentially beneficial area of improvement is the extension of node capabilities considered by the reviewed work.

With the growing adoption of cyber-physical systems, the awareness of physical node capabilities becomes increasingly important. For example, attempting to request a live video feed from a node with no camera attached is pointless. Therefore, it may be technically impossible to achieve new system goals with the existing resources. These complications lead to the necessity of more research to be conducted in the area of run-time system mission-related changes to develop more robust and flexible system resilience orchestration.

FIGURE 9. System resilience orchestration components.

VII. CHALLENGES AND OPPORTUNITIES

At a higher abstraction level, achieving system resilience requires addressing multiple system-wide aspects such as formal resilience metrics, capability-aware resource management and mission definition. At a lower abstraction level, individual node communication must be extended to take individual node capabilities into consideration. Enhancing currently existing cyber-foraging techniques to acknowledge physical node capabilities during the utility function calculation can be highly beneficial in IoT scenarios. Further details on the related challenges are discussed in this section.

A. FORMAL RESILIENCE ORCHESTRATION

System resilience is commonly defined in terms of a system's ability to provide an ''acceptable level of operation'' under certain conditions [105]. Unfortunately, strict resilience metrics are not universally defined [106], [132], [142]. Once the operation level degrades below a pre-set threshold, various restoration procedures must be carried out until the desired level of operation is achieved. Large-scale cloud systems might require additional steps in order to pinpoint the exact location of a fault [107].

An efficient system resilience orchestration must be derived from a formal system node capabilities and system tasks definitions (Figure 9). At a higher level, the resilience orchestration can be perceived as a continuous two-phase process, analysis phase and action phase. The analysis phase is active and is responsible for three main tasks:

- Determining whether the current system goals are achievable using currently available resources;
- Locating critical system nodes (those possessing mission-critical capabilities);
- Providing investment recommendations to make a system goals reachable.

The orchestration switches to the action phase in case changes in either a system's requirements or the available node capabilities occur during a system's operation. The purpose of the action phase is to:

TABLE 2. Resilience solutions summary.

Resilience approach	Efficiency	Capacity	Trustworthiness	Service level	Protocol level	Network level	Resilience approach	Efficiency	Capacity	Trustworthiness	Service level	Protocol level	Network level
Abbasi, H. et al., 2010 [102]		✓		✓			Laud, P. & Pankova, A., 2014 [45]		✓	✓	✓	✓	
Abdul-Rahman, A. & Hailes, S., 1998 [41]			✓		✓		Lewis, G. A. et al., 2014 [70]		✓		✓		
Ahnn, J. H. & Potkonjak, M., 2013 [75]	✓	✓				✓	Lewis, G. et al., 2014 [22]		✓		✓		
Al-Ayyoub, et al., 2018 [13]	✓	✓		✓			Li, H. & Singhal, M., 2007 [42]			✓		✓	
Alcaraz, C., 2018 [131]	✓					✓	Liu, Y., Li, X. & Xiao, L., 2018 [141]	✓			✓		
Alcaraz, et al., 2017 [65]	✓	✓				✓	Lu, L. et al., 2008 [43]			✓	✓	✓	
Babaoglu, O., Marzolla, M. & Tamburini, M., 2012 [116]	✓					✓	Matos, D. R. et al., 2018 [139]			✓	✓		
Balan, R. et al., 2002 [67]	✓		✓	✓			Matthews, J. et al., 2011 [113]		✓		✓		
Balan, R. K. et al., 2007 [39]	✓			✓			Menascé, D. A., 2002 [126]	✓			✓		
Bao, F. & Chen, I.-R., 2012 [58]			✓		✓		Menth, M. & Martin, R., 2005 [89]		✓				✓
Benson, K. E., et al., 2018 [145]	✓					✓	Mertoguno, J. S. et al., 2019 [138]	✓			✓		
Benson, K., 2015 [98]	✓				✓	✓	Mohr, A. E., Riskin, E. A. & Ladner, R. E., 2000 [91]		✓			✓	
Burns, et al., 2018 [90]		✓			✓		Najjar, W. & Gaudiot, J.-L., 1990 [87]		✓				✓
Canetti, R., Riva, B. & Rothblum, G. N., 2011 [47]			✓	✓			Neisse, R. et al., 2015 [54]			✓	✓		
Chang, Y.-S. & Hung, S.-H., 2011 [112]	✓		✓	✓			Osanaiye, O., Choo, K.-K. R. & Dlodlo, M., 2016 [16]	✓			✓		
Chen, D., Garg, S. & Trivedi, K. S., 2002 [85]	✓					✓	Parno, B., et al., 2013 [48]		✓	✓	✓		
Chen, M. et al., 2015 [19]	✓					✓	Prabhakar, R. et al., 2011 [103]	✓	✓				✓
Chu, H.-h.et al., 2004 [111]	✓			✓			Satyanarayanan, M. et al., 2009 [61]		✓		✓		
Damiani, E. et al., 2002 [118]			✓	✓			Satyanarayanan, M. et al., 2013 [10]	✓	✓		✓		
Di, S. & Wang, C.-L., 2013a [58]	✓	✓		✓		✓	Senejohnny, D. M. et al., 2019 [137]			✓		✓	
Di, S. & Wang, C.-L., 2013b [69]	✓			✓		✓	Shelton, C. P., Koopman, P. & Nace, W., 2003 [95]	✓			✓		
Dialani, V. et al., 2002 [93]	✓			✓			Shi, E. & Perrig, A., 2004 [97]			✓	✓		✓
Dubey, A., Karsai, G. & Pradhan, S., 2017 [35]	✓				✓	✓	Simmhan, Y. et al., 2009 [68]	✓			✓		
Dustdar, S. & Juszczyk, L., 2007 [128]	✓			✓			Sun, X., Liu, P. & Singhal, A., 2018 [136]	✓			✓		
Echeverria, S. et al., 2015 [20]	✓	✓		✓			Sun, Y. et al., 2016 [62]	✓			✓		
Flinn, J. et al., 2003 [49]	✓		✓	✓			Villarreal-Vasquez, et al., 2017 [44]			✓	✓		
Florins, M. & Vanderdonckt, J., 2004 [94]	✓	✓		✓			Vishwanath, A., Peruri, R. & He, J. S., 2016 [51]			✓		✓	
Garraghan, P. et al., 2018 [144]	✓			✓			Wang, H., Shen, H. & Li, Z., 2018 [134]	✓	✓		✓		
Gennaro, R., Gentry, C. & Parno, B., 2010 [46]	✓		✓	✓			Wang, J. et al., 2019 [133]	✓					✓
Goyal, S. & Carter, J., 2004 [71]	✓	✓	✓	✓			Wu, Y. et al., 2018 [132]	✓			✓		
Ha, K., et al., 2011 [9]	✓	✓	✓			✓	Yang, C.-S. & Luo, M.-Y., 2000 [92]	✓	✓		✓		
Hada, P. S., Singh, R. & Manmohan, M., 2011 [15]			✓	✓			Yang, Z. et al., 2010 [18]		✓		✓		
Javaid, S. et al., 2019 [143]			✓	✓			Zhang, M. et al., 2019 [130]	✓					✓
Keller, L., Upadhyaya, P. & Candea, G., 2008 [96]		✓		✓			Zhang, X. et al., 2011 [74]		✓		✓		
Kemp, R. et al., 2012 [110]	✓	✓	✓	✓			Zhang, Z. et al., 2007 [104]	✓	✓		✓		
Kosta, S. et al., 2012 [81]		✓		✓			Zhao, P. et al., 2010 [122]	✓					✓
Kwon, Y.-W. & Tilevich, E., 2013 [77]		✓		✓									

- Carry out recovery procedures in case the defined system requirements cannot be met with the current system nodes configuration;
- Adapt to the changes in the available resources and a system's goals;
- Report failures when system recovery is not possible.

Capability-based architecture is also useful in handling traditional cyber-foraging and reverse foraging in a uniform manner. For example, a command center may request the available drones to send their video feeds to a powerful surrogate node for further processing. Video processing results are then sent back to the command center, which may take a decision to order certain nodes to attack the detected target. The order of actions in this process is dictated by the capabilities of the participating nodes and a system's mission. Once the basic system operation is achieved with the existing resources and capabilities, a node's dynamic capability can be used for reconfiguration to improve a system resilience and reliability.

Due to various run-time events, a system's service level may severely degrade. A resilient system may either make disastrous events unlikely (through redundancy) or minimize the impact of such events (run-time adaptation). With the advances of large distributed networks consisting of mobile devices, it has become possible to combine both approaches for providing redundancy for the selected nodes at run-time based on a system's state.

In addition to the external events affecting a system's operation level, the nature of a system's mission itself may change at run-time. To achieve such level of flexibility, the proposed framework is expected to allow dynamic changes to the system's workflow definition. Whilst partial algorithm and protocol level implementations are proposed for some system architectures and specific scenarios [108], a universal extensible approach is yet to be developed. An integral part of the proposed framework is the notion of node capabilities, which can be used to drive individual node communications.

B. CAPABILITY-AWARE CYBER-FORAGING

A common assumption behind cyber-foraging approaches is that end-point nodes and surrogate nodes only differ in terms

of capacity [39]. Offloading data is performed because a surrogate node has more disk space; offloading of computation is expected to utilize more processing power available to surrogate nodes and saving battery life of an end-point node.

This simplistic view of cyber-foraging can be extended to consider more aspects such as node roles and capabilities. For example, an end device may communicate with a remote server not because of lack of computational resources or other capabilities, but because the remote server possesses some important information. In this scenario, a remote server has a different role (authority), leading to the necessity for end-point devices to send requests to a server. Complex cyber-physical systems such as military equipment distributed across a certain territory may contain nodes vastly differing in their physical capabilities (and roles). Therefore, cyber-foraging decisions must be taken based on the available capabilities in addition to the available capacity.

One of the commonly raised issues is the calculation of offload choices (known as utility function) [109]. As opposed to always offloading, some computations to a nearest cloudlet, it may become more effective to perform the computations locally under certain conditions. Therefore, the proposed approaches tend to make this type of decisions automatically at run time [110]–[112]. However, the underlying benchmarks are generally taken under laboratory conditions and may not necessarily reflect the real-world usage scenarios [79]. Thus, defining an optimal utility function suitable for different environments is still largely an unsolved problem. Some approaches may, therefore, take offloading choice based on user preferences [81], [113].

Whilst a multitude of significant factors such as battery charge level and communication delays are commonly taken into consideration, perhaps in efforts to simplify the utility function computation, some approaches tend to dismiss some seemingly less important factors.

Notable examples include encryption-related delays, the battery charge required to maintain the connectivity whilst waiting for a response and the time required to compute the value of the utility function itself. However, taking such secondary factors into consideration may become useful. The factors commonly assessed by cyber-foraging utility functions are presented in Figure 10.

The reviewed literature predominantly associates cyber-foraging only with the computational capabilities of surrogate nodes. Physical node capability-aware resilience is currently somewhat under-researched. Some existing works completely overlook device capabilities [146]. Achieving resilience in both IoT and CPS (Cyber-Physical Systems) domains is typically done on node level [138], [140]. Distinctive features of CPS and IoT systems relevant to resilience include service-critical device capabilities. In such systems, different hardware failures may have vastly different outcomes in terms of system operation. For instance, in contrast to worsened network conditions, a failed video camera may render the whole video surveillance system ineffective. This

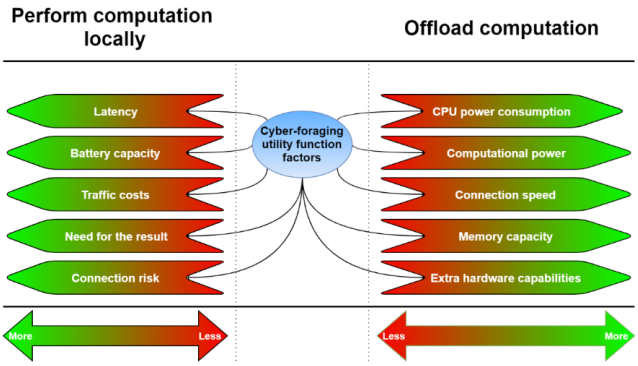

FIGURE 10. Factors affecting cyber-foraging decisions.

significantly complicates automated system operation recovery and future adaptation for complex CPS and IoT systems.

A considerable number of CPS and IoT resilience studies focus on data exchange, without taking physical data sources into consideration [131], [137], [145]. Notable exceptions include focusing on physical environment-affecting node capabilities in order to evaluate overall system safety in CPS domain [135]. Due to the high importance of data typically circulating within IoT and CPS systems, resilience against malicious nodes also gains significant attention [143].

Perhaps extending a semantic web-oriented ontology language such as OWL [114] to focus on physical device capabilities would potentially be beneficial.

VIII. CONCLUSION

Our review has enabled us to conclude that there has been more research on the software systems resilience compared with the resilience of cyber-physical systems. There are several well researched and deployed techniques such as load-balancing, failover nodes, node discovery and centralized trust management. However, the rapidly rising adoption of IoT-based systems, which heavily rely on the diversity of physical nodes' capabilities, poses challenges leading to more cyber-physical resilience-oriented gaining attention [138], [147].

There is an important need of gaining a deeper understanding of nodes capabilities and roles for efficient resilience orchestration in modern cyber-physical distributed systems. Similarly an increased attention is being paid to power requirements and mobility aspects. It is likely that more and more devices' characteristics will become increasingly important with the adoption of IoT. Thus, a more structured and extensible formal approaches need to be developed to achieve highly flexible and capability-aware system resilience orchestration. Such improvements can be beneficial in scenarios involving a multitude of extremely different nodes such as military and rescue operations.

There is a need of further research in the area of trustworthy collaborations among different systems. Dynamic systems comprising of nodes belonging to different parties (perhaps not fully trusting each other) raise additional trust- and security-related questions to be answered.

REFERENCES

[1] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. 5th Int. Joint Conf. INC, IMS IDC*, 2009, pp. 44–51.

[2] 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," Dept. EECS, Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009, vol. 17.

[3] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[4] R. K. Lomotey and R. Deters, "Architectural designs from mobile cloud computing to ubiquitous cloud computing–survey," in *Proc. IEEE World Congr. Services*, Jun. 2014, pp. 418–425.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[6] *OpenFog Architecture Overview*, OpenFog Consortium Architecture Working Group, Fremont, CA, USA, Feb. 2016.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2017, pp. 1–6.

[9] K. Ha, G. Lewis, S. Simanta, and M. Satyanarayanan, *Cloud Offload in Hostile Environments*. Pittsburgh, PA, USA: School of Computer ScienceCarnegie Mellon Univ., 2011.

[10] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervas. Comput.*, vol. 12, no. 4, pp. 40–49, Oct. 2013.

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

[12] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Proc. Int. Conf. Comput., Electron. Electr. Technol. (ICCEET)*, 2012, pp. 877–880.

[13] M. Al-Ayyoub, M. Al-Quraan, Y. Jararweh, E. Benkhelifa, and S. Hariri, "Resilient service provisioning in cloud based data centers," *Future Gener. Comput. Syst.*, vol. 86, pp. 765–774, Sep. 2018.

[14] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci.*, Nov. 2010, pp. 693–702.

[15] P. S. Hada, R. Singh, and M. Manmohan, "Security agents: A mobile agent based trust model for cloud computing," *Int. J. Comput. Appl.*, vol. 36, pp. 12–15, Dec. 2011.

[16] O. Osanaiye, K.-K.-R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework," *J. Netw. Comput. Appl.*, vol. 67, pp. 147–165, May 2016.

[17] R. Jhawar and V. Piuri, "Fault tolerance and resilience in cloud computing environments," in *Computer and Information Security Handbook*. New York, NY, USA: Morgan Kaufmann, 2013, pp. 125–141.

[18] Z. Yang, B. Y. Zhao, Y. Xing, S. Ding, F. Xiao, and Y. Dai, "AmazingStore: Available, low-cost online storage service using cloudlets." in *Proc. IPTPS*, vol. 10, 2010, p. 2.

[19] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 18–24, Jun. 2015.

[20] S. Echeverria, G. A. Lewis, J. Root, and B. Bradshaw, "Cyber-foraging for improving survivability of mobile systems," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2015, pp. 1421–1426.

[21] L. Tawalbeh, N. Alassaf, W. Bakheder, and A. Tawalbeh, "Resilience mobile cloud computing: Features, applications and challenges," in *Proc. 5th Int. Conf. e-Learn. (ECONF)*, Oct. 2015, pp. 280–284.

[22] G. Lewis, S. Echeverria, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *Proc. IEEE Military Commun. Conf.*, Oct. 2014, pp. 1440–1446.

[23] M. Firdhous, O. Ghazali, and S. Hassan, "Fog computing: Will it be the future of cloud computing?" in *Proc. 3rd Int. Conf. Informat. Appl. (ICIA)*, 2014, pp. 1–8.

[24] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[25] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

[26] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[27] K. A. Delic, "On resilience of IoT systems: The Internet of Things (ubiquity symposium)," *Ubiquity*, vol. 2016, pp. 1–7, Feb. 2016.

[28] F. Al-Doghman, Z. Chaczko, A. R. Ajayan, and R. Klempous, "A review on fog computing technology," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 1525–1530.

[29] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog *et al.*: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[30] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-A key technology towards 5G," *ETSI White Paper*, vol. 11, pp. 1–16, 2015.

[31] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. 10th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2016, pp. 1–8.

[32] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. Int. Conf. Adv. Future Internet (AFIN)*, Lisbon, Portugal, Nov. 2014, pp. 48–54.

[33] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Proc. WASA*, 2015, pp. 685–695.

[34] F. Popentiu-Vladicescu and G. Albeanu, "Software reliability in the fog computing," in *Proc. Int. Conf. Innov. Electr. Eng. Comput. Technol. (ICIEECT)*, Apr. 2017, pp. 1–4.

[35] A. Dubey, G. Karsai, and S. Pradhan, "Resilience at the edge in cyber-physical systems," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 139–146.

[36] R. Jhawar and V. Piuri, "Fault tolerance and resilience in cloud computing environments," in *Computer and Information Security Handbook*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2017, pp. 165–181.

[37] M. Stoicescu, J.-C. Fabre, and M. Roy, "Architecting resilient computing systems: A component-based approach for adaptive fault tolerance," *J. Syst. Archit.*, vol. 73, pp. 6–16, Feb. 2017.

[38] P. Huang, C. Guo, L. Zhou, J. R. Lorch, Y. Dang, M. Chintalapati, and R. Yao, "Gray failure: The achilles' heel of cloud-scale systems," in *Proc. 16th Workshop Hot Topics Operating Syst.*, 2017, pp. 150–155.

[39] R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb, "Simplifying cyber foraging for mobile devices," in *Proc. 5th Int. Conf. Mobile Syst., Appl. Services*, 2007, pp. 272–285.

[40] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[41] A. Abdul-Rahman and S. Hailes, "A distributed trust model," in *Proc. Workshop New Secur. Paradigms*, 1997, pp. 48–60.

[42] H. Li and M. Singhal, "Trust management in distributed systems," *Computer*, vol. 40, no. 2, pp. 45–53, Feb. 2007.

[43] L. Lu, J. Han, Y. Liu, L. Hu, J.-P. Huai, L. Ni, and J. Ma, "Pseudo trust: Zero-knowledge authentication in anonymous P2Ps," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 10, pp. 1325–1337, Oct. 2008.

[44] M. Villarreal-Vasquez, B. Bhargava, P. Angin, N. Ahmed, D. Goodwin, K. Brin, and J. Kobes, "An MTD-based self-adaptive resilience approach for cloud systems," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 723–726.

[45] P. Laud and A. Pankova, "Verifiable computation in multiparty protocols with honest majority," Cryptol. ePrint Arch., Tech. Rep. 2014/060, 2014. [Online]. Available: http://eprint.iacr.org

[46] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology-CRYPTO 2010*, 2010, pp. 465–482.

[47] R. Canetti, B. Riva, G. Rothblum, "Verifiable computation with two or more clouds," in *Proc. Workshop Cryptogr. Secur. Clouds*, Zürich, Switzerland, Mar. 2011.

[48] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 238–252.

[49] J. Flinn, S. Sinnamohideen, N. Tolia, and M. Satyanarayanan, "Data staging on untrusted surrogates," *Proc. FAST*, vol. 3, pp. 15–28, 2003.

[50] M. Alizadeh, S. Abolfazli, M. Zamani, S. Baharun, and K. Sakurai, "Authentication in mobile cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 61, pp. 59–80, Feb. 2016.
[51] A. Vishwanath, R. Peruri, and J. He, "Security in fog computing through encryption," *Int. J. Inf. Technol. Comput. Sci.*, vol. 8, no. 5, pp. 28–36, May 2016.
[52] Z. Priscakova and I. Rabova, "Model of solutions for data security in cloud computing," 2013, *arXiv:1307.3766*. [Online]. Available: https://arxiv.org/abs/1307.3766
[53] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.
[54] R. Neisse, G. Steri, I. N. Fovino, and G. Baldini, "SecKit: A model-based security toolkit for the Internet of Things," *Comput. Secur.*, vol. 54, pp. 60–76, Oct. 2015.
[55] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2014, pp. 1–8.
[56] L. Rudolph, "A virtualization infrastructure that supports pervasive computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 8–13, Oct. 2009.
[57] M. H. Ibrahim, "Octopus: An edge-fog mutual authentication scheme," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1089–1101, Nov. 2016.
[58] F. Bao and I.-R. Chen, "Dynamic trust management for Internet of Things applications," in *Proc. Int. Workshop Self-Aware Internet Things*, 2012, pp. 1–6.
[59] S. Di and C.-L. Wang, "Error-tolerant resource allocation and payment minimization for cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1097–1106, Jun. 2013.
[60] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Comput. Netw.*, vol. 54, no. 8, pp. 1245–1265, Jun. 2010.
[61] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.
[62] Y. Sun, J. White, S. Eade, and D. C. Schmidt, "ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization," *J. Syst. Softw.*, vol. 116, pp. 146–161, Jun. 2016.
[63] A. Schaeffer-Filho, P. Smith, A. Mauthe, D. Hutchison, Y. Yu, and M. Fry, "A framework for the design and evaluation of network resilience management," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 401–408.
[64] P. Smith, A. Schaeffer-Filho, A. Ali, M. Schöller, N. Kheir, A. Mauthe, and D. Hutchison, "Strategies for network resilience: Capitalising on policies," in *Proc. AIMS*, pp. 118–122, 2010.
[65] C. Alcaraz, J. Lopez, and K.-K.-R. Choo, "Resilient interconnection in cyber-physical control systems," *Comput. Secur.*, vol. 71, pp. 2–14, Nov. 2017.
[66] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci.*, vol. 8, no. 4, pp. 1–10, Apr. 2016.
[67] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, "The case for cyber foraging," in *Proc. 10th Workshop ACM SIGOPS Eur. Workshop, Beyond PC (EW10)*, New York, NY, USA, 2002, pp. 87–92.
[68] Y. Simmhan, R. Barga, C. van Ingen, M. Nieto-Santisteban, L. Dobos, N. Li, M. Shipway, A. S. Szalay, S. Werner, and J. Heasley, "GrayWulf: Scalable software architecture for data intensive computing," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci.*, 2009, pp. 1–10.
[69] S. Di and C.-L. Wang, "Dynamic optimization of multiattribute resource allocation in self-organizing clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 464–478, Mar. 2013.
[70] G. A. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments," in *Proc. Companion Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 412–415.
[71] S. Goyal and J. Carter, "A lightweight secure cyber foraging infrastructure for resource-constrained devices," in *Proc. 6th IEEE Workshop Mobile Comput. Syst. Appl.*, Apr. 2005, pp. 186–195.
[72] S.-H. Hung, J.-P. Shieh, and C.-P. Lee, "Migrating android applications to the cloud," *Int. J. Grid High Perform. Comput.*, vol. 3, no. 2, pp. 14–28, 2011, doi: 10.4018/jghpc.2011040102.
[73] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Netw. Appl.*, vol. 16, no. 3, pp. 270–284, Jun. 2011.
[74] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.
[75] J. H. Ahnn and M. Potkonjak, "mHealthMon: Toward energy-efficient and distributed mobile health monitoring using parallel offloading," *J. Med. Syst.*, vol. 37, p. 1, Oct. 2013.
[76] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2009, pp. 280–293.
[77] Y.-W. Kwon and E. Tilevich, "Reducing the energy consumption of mobile applications behind the scenes," in *Proc. IEEE Int. Conf. Softw. Maintenance*, Sep. 2013, pp. 170–179.
[78] G. A. Lewis and P. Lago, "A catalog of architectural tactics for cyber-foraging," in *Proc. 11th Int. ACM SIGSOFT Conf. Qual. Softw. Archit.*, 2015, pp. 53–62.
[79] G. Lewis and P. Lago, "Architectural tactics for cyber-foraging: Results of a systematic literature review," *J. Syst. Softw.*, vol. 107, pp. 158–186, Sep. 2015.
[80] G. A. Lewis, "Software architecture strategies for cyber-foraging systems," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2016.
[81] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.
[82] B. Green, M. Krotofil, and D. Hutchison, "Achieving ICS resilience and security through granular data flow management," in *Proc. 2nd ACM Workshop Cyber-Phys. Syst. Secur. Privacy*, 2016, pp. 93–101.
[83] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. Jones, "A survey of cyber security management in industrial control systems," *Int. J. Crit. Infrastruct. Protect.*, vol. 9, pp. 52–80, Jun. 2015.
[84] S. Mittal and M. S. Inukonda, "A survey of techniques for improving error-resilience of DRAM," *J. Syst. Archit.*, vol. 91, pp. 11–40, Nov. 2018.
[85] D. Chen, S. Garg, and K. S. Trivedi, "Network survivability performance evaluation: A quantitative approach with applications in wireless ad-hoc networks," in *Proc. 5th ACM Int. Workshop Modeling Anal. Simulation Wireless Mobile Syst.*, 2002, pp. 61–68.
[86] A. Binzenhöfer and K. Leibnitz, "Estimating churn in structured P2P networks," in *Proc. 20th Int. Teletraffic Conf. Manag. Traffic Perform. Converged Netw.* Springer-Verlag, 2007, pp. 630–641.
[87] W. Najjar and J.-L. Gaudiot, "Network resilience: A measure of network fault tolerance," *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 174–181, Feb. 1990.
[88] J. P. G. Sterbenz, E. K. Çetinkaya, M. A. Hameed, A. Jabbar, S. Qian, and J. P. Rohrer, "Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation," *Telecommun. Syst.*, vol. 52, no. 2, pp. 705–736, Feb. 2013.
[89] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *Proc. 5th Int. Workshop Design Reliable Commun. Netw. (DRCN)*, Jan. 2006, pp. 271–277.
[90] A. Burns, J. Harbin, L. Indrusiak, I. Bate, R. Davis, and D. Griffin, "AirTight: A resilient wireless communication protocol for mixed-criticality systems," in *Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2018, pp. 65–75.
[91] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 819–828, Jun. 2000.
[92] C.-S. Yang and M.-Y. Luo, "Realizing fault resilience in Web-server cluster," in *Proc. ACM/IEEE SC Conf. (SC)*, 2000, p. 21.
[93] V. Dialani, S. Miles, L. Moreau, D. De Roure, and M. Luck, "Transparent fault tolerance for Web services based architectures," *Euro-Par*, vol. 2, pp. 889–898, Aug. 2002.
[94] M. Florins and J. Vanderdonckt, "Graceful degradation of user interfaces as a design method for multiplatform systems," in *Proc. 9th Int. Conf. Intell.*, vol. 4, 2004, pp. 140–147.

[95] C. Shelton, P. Koopman, and W. Nace, "A framework for scalable analysis and design of system-wide graceful degradation in distributed embedded systems," in *Proc. 8th Int. Workshop Object-Oriented Real-Time Dependable Syst. (WORDS)*, Oct. 2003, pp. 156–163.
[96] L. Keller, P. Upadhyaya, and G. Candea, "ConfErr: A tool for assessing resilience to human configuration errors," in *Proc. IEEE Int. Conf. Dependable Syst. Netw. FTCS DCC (DSN)*, Jun. 2008, pp. 157–166.
[97] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Commun.*, vol. 11, no. 6, pp. 38–43, Dec. 2004.
[98] K. Benson, "Enabling resilience in the Internet of Things," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2015, pp. 230–232.
[99] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 337–368, 1st Quart., 2014.
[100] M. Viroli, G. Audrito, J. Beal, F. Damiani, and D. Pianini, "Engineering resilient collective adaptive systems by self-stabilisation," *ACM Trans. Model. Comput. Simul.*, vol. 28, no. 2, pp. 1–28, Mar. 2018.
[101] D. Gambetta "'Can we trust trust?"" in *Trust: Making and Breaking Cooperative Relations*, D. Gambetta, Ed, Electron. ed. Oxford, U.K.: Department of Sociology, Univ. Oxford, 2000, ch. 13, pp. 213–237.
[102] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "DataStager: Scalable data staging services for petascale applications," *Cluster Comput.*, vol. 13, no. 3, pp. 277–290, Sep. 2010.
[103] R. Prabhakar, S. S. Vazhkudai, Y. Kim, A. R. Butt, M. Li, and M. Kandemir, "Provisioning a multi-tiered data staging area for extreme-scale machines," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 1–12.
[104] Z. Zhang, C. Wang, S. S. Vazhkudai, X. Ma, G. G. Pike, J. W. Cobb, and F. Mueller, "Optimizing center performance through coordinated data staging, scheduling and recovery," in *Proc. ACM/IEEE Conf. Supercomput.*, 2007, p. 55.
[105] P. Smith, D. Hutchison, J. P. Sterbenz, M. Schöller, A. Fessi, M. Karaliopoulos, C. Lac, and B. Plattner, "Network resilience: A systematic approach," *IEEE Commun. Mag.*, vol. 49, no. 7, pp. 88–97, Jul. 2011.
[106] M. Novak, S. N. Shirazi, A. Hudic, T. Hecht, M. Tauber, D. Hutchison, S. Maksuti, and A. Bicaku, "Towards resilience metrics for future cloud applications," in *Proc. 6th Int. Conf. Cloud Comput. Services Sci.*, vol. 1, 2016, pp. 295–301.
[107] L. Mariani, C. Monni, M. Pezze, O. Riganelli, and R. Xin, "Localizing faults in cloud systems," in *Proc. IEEE 11th Int. Conf. Softw. Test., Verification Validation (ICST)*, Apr. 2018, pp. 262–273.
[108] A. Marnerides, C. James, A. S. Filho, S. Y. Sait, A. Mauthe, and H. Murthy, "Multi-level network resilience: traffic analysis, anomaly detection and simulation," *J. Commun. Technol., Special Issue Next Gener. Wireless Netw. Appl.*, vol. 2, no. 2, pp. 345–356, Jun. 2011.
[109] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[110] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. Mobile Comput., Appl., Services, 2nd Int. ICST Conf. (MobiCASE)*, M. Gris and G. Yang, Eds. Berlin, Germany: Springer, 2012, pp. 59–79.
[111] H.-H. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri, "Roam, a seamless application framework," *J. Syst. Softw.*, vol. 69, no. 3, pp. 209–226, Jan. 2004.
[112] Y.-S. Chang and S.-H. Hung, "Developing collaborative applications with mobile cloud-a case study of speech recognition," *J. Internet Services Inf. Secur.*, vol. 1, pp. 18–36, 2011.
[113] J. Matthews, M. Chang, Z. Feng, R. Srinivas, and M. Gerla, "PowerSense: Power aware dengue diagnosis on mobile phones," in *Proc. 1st ACM Workshop Mobile Syst., Appl., Services Healthcare*, 2011, p. 6.
[114] S. Bechhofer, "OWL: Web ontology language," in *Encyclopedia of Database Systems*. Boston, MA, USA: Springer, 2009, pp. 2008–2009.
[115] A. Bahtovski and M. Gusev, "Cloudlet Challenges," *Procedia Eng.*, vol. 69, pp. 704–711, 2014.
[116] O. Babaoglu, M. Marzolla, and M. Tamburini, "Design and implementation of a P2P Cloud system," in *Proc. 27th Annu. ACM Symp. Appl. Comput.*, 2012, pp. 412–417.
[117] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos, "Highly resilient peer-to-peer botnets are here: An analysis of Gameover Zeus," in *Proc. 8th Int. Conf. Malicious Unwanted Softw.*, Oct. 2013, pp. 116–123.
[118] E. Damiani, D. C. Di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proc. 9th ACM Conf. Comput. Commun. Secur.*, 2002, pp. 207–216.
[119] R. Choubey, R. Dubey, and J. Bhattacharjee, "A survey on cloud computing security, challenges and threats," *Int. J. Comput. Sci. Eng.*, vol. 3, pp. 1227–1231, Nov. 2011.
[120] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.
[121] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of Fog computing and its security issues," *Concurrency Computat., Pract. Exper.*, vol. 28, no. 10, pp. 2991–3005, Jul. 2016.
[122] P. Zhao, T.-L. Huang, C.-X. Liu, and X. Wang, "Research of P2P architecture based on cloud computing," in *Proc. Int. Conf. Intell. Comput. Integr. Syst.*, Oct. 2010, pp. 652–655.
[123] N.-U.-H. Shirazi, S. Simpson, S. Oechsner, A. Mauthe, and D. Hutchison, "A framework for resilience management in the cloud," *Elektrotech. Inftech.*, vol. 132, no. 2, pp. 122–132, Mar. 2015.
[124] W. Zhao, "BFT-WS: A byzantine fault tolerance framework for Web services," in *Proc. 11th Int. IEEE EDOC Conf. Workshop*, Oct. 2007, pp. 89–96.
[125] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms*. Upper Saddle River, NJ, USA: Prentice-Hall, 2007.
[126] D. Menasce, "QoS issues in Web services," *IEEE Internet Comput.*, vol. 6, no. 6, pp. 72–75, Nov. 2002.
[127] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.
[128] S. Dustdar and L. Juszczyk, "Dynamic replication and synchronization of Web services for high availability in mobile ad-hoc networks," *Service Oriented Comput. Appl.*, vol. 1, no. 1, pp. 19–33, Apr. 2007.
[129] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2586–2595, Nov. 2017.
[130] M. Zhang, J. Bi, K. Gao, Y. Qiao, G. Li, X. Kong, Z. Li, and H. Hu, "Tripod: Towards a scalable, efficient and resilient cloud gateway," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 570–585, Mar. 2019.
[131] C. Alcaraz, "Cloud-assisted dynamic resilience for cyber-physical control systems," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 76–82, Feb. 2018.
[132] Y. Wu, Y. Fang, B. Liu, and Z. Zhao, "A novel service deployment approach based on resilience metrics for service-oriented system," *Pers. Ubiquit Comput.*, vol. 22, nos. 5–6, pp. 1099–1107, Oct. 2018.
[133] J. Wang, S. Pambudi, W. Wang, and M. Song, "Resilience of IoT systems against edge-induced cascade-of-failures: A networking perspective," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6952–6963, Aug. 2019.
[134] H. Wang, H. Shen, and Z. Li, "Approaches for resilience against cascading failures in cloud datacenters," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 706–717.
[135] S. A. Timashev, "Cyber reliability, resilience, and safety of physical infrastructures," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 481, Mar. 2019, Art. no. 012009.
[136] X. Sun, P. Liu, and A. Singhal, "Toward cyberresiliency in the context of cloud computing [resilient security]," *IEEE Secur. Privacy*, vol. 16, no. 6, pp. 71–75, Nov. 2018.
[137] D. M. Senejohnny, S. Sundaram, C. De Persis, and P. Tesi, "Resilience against misbehaving nodes in asynchronous networks," *Automatica*, vol. 104, pp. 26–33, Jun. 2019.
[138] J. S. Mertoguno, R. M. Craven, M. S. Mickelson, and D. P. Koller, "A physics-based strategy for cyber resilience of CPS," in *Proc. SPIE, Auton. Syst., Sensors, Process. Secur. Vehicles Infrastruct.*, Baltimore, MD, USA, vol. 11009, May 2019, Art. no. 110090E, doi: 10.1117/12.2517604.
[139] D. R. Matos, M. L. Pardal, G. Carle, and M. Correia, "RockFS: Cloud-backed file system resilience to client-side attacks," in *Proc. 19th Int. Middleware Conf.*, New York, NY, USA, 2018, pp. 107–119.

[140] A. Lukina, A. Tiwari, S. A. Smolka, L. Esterle, J. Yang, and R. Grosu, "Resilient control and safety for cyber-physical systems," in *Proc. IEEE Workshop Monitor. Test. Cyber-Phys. Syst. (MT-CPS)*, Apr. 2018, pp. 16–17.
[141] Y. Liu, X. Li, and L. Xiao, "Service oriented resilience strategy for cloud data center," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2018, pp. 269–274.
[142] I. Linkov and A. Kott, "Fundamental concepts of cyber resilience: Introduction and overview," in *Cyber Resilience of Systems and Networks*, I. Linkov and A. Kott, Eds. Champlain, IL, USA: Springer, 2019, pp. 1–25.
[143] S. Javaid, H. Afzal, M. Babar, F. Arif, Z. Tan, and M. A. Jan, "ARCA-IoT: An attack-resilient cloud-assisted IoT system," *IEEE Access*, vol. 7, pp. 19616–19630, 2019.
[144] P. Garraghan, R. Yang, Z. Wen, A. Romanovsky, J. Xu, R. Buyya, and R. Ranjan, "Emergent failures: Rethinking cloud reliability at scale," *IEEE Cloud Comput.*, vol. 5, no. 5, pp. 12–21, Sep. 2018.
[145] K. E. Benson, G. Wang, N. Venkatasubramanian, and Y.-J. Kim, "Ride: A resilient IoT data exchange middleware leveraging SDN and edge cloud resources," in *Proc. IEEE/ACM 3rd Int. Conf. Internet-Things Design Implement. (IoTDI)*, Apr. 2018, pp. 72–83.
[146] R. Alguliyev, Y. Imamverdiyev, and L. Sukhostat, "Cyber-physical systems and their security issues," *Comput. Ind.*, vol. 100, pp. 212–223, Sep. 2018.
[147] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A survey," *ACM Trans. Internet Technol.*, vol. 19, no. 2, p. 18, 2019.

VICTOR PROKHORENKO received the Ph.D. degree in computer science from the University of South Australia. He is currently a Researcher with the Centre for Research on Engineering Software Technologies (CREST), The University of Adelaide. He has more than 14 years of experience in software engineering with main areas of expertise including investigation of technologies related to software resilience, trust management, and big data solutions hosted within OpenStack private cloud platform.

M. ALI BABAR is currently a Professor with the School of Computer Science, The University of Adelaide. He is an Honorary Visiting Professor with the Software Institute, Nanjing University, China. He is the Director of Cyber Security Adelaide (CSA), which incorporates a node of recently approved the Cyber Security Cooperative Research Centre (CSCRC), whose estimated budget is around A$140 Millions over seven years with A$50 Millions provided by the Australia Government. In the area of Software Engineering education, he led the University's effort to redevelop a Bachelor of Engineering (software) degree that has been accredited by the Australian Computer Society and the Engineers Australia (ACS/EA). Prior to joining The University of Adelaide, he spent almost seven years in Europe (Ireland, Denmark, and U.K.) working as a Senior Researcher and an Academic. Before returning to Australia, he was a Reader of software engineering with Lancaster University. He has established an Interdisciplinary Research Centre, Centre for Research on Engineering Software Technologies (CREST), where he leads the research and research training of more than 20 (12 Ph.D. students) members. Apart from his work having industrial relevance as evidenced by several R&D projects and setting up a number of collaborations in Australia and Europe with industry and government agencies, his publications have been highly cited within the discipline of Software Engineering as evidenced by his H-index is 46 with 8240 citations as per Google Scholar on January 20, 2020. He leads the theme on Platform and Architecture for Cyber Security as a Service with the Cyber Security Cooperative Research Centre. He has authored/coauthored more than 220 peer-reviewed publications through premier Software Technology journals and conferences.