Research Challenges in Nextgen Service Orchestration

Luis M. Vaquero, Felix Cuadrado, Yehia Elkhatib, Jorge Bernal-Bernabe, Satish N. Srirama, Mohamed Faten Zhani

Abstract

Fog/edge computing, function as a service, and programmable infrastructures, like software-defined networking or network function virtualisation, are becoming ubiquitously used in modern Information Technology infrastructures. These technologies change the characteristics and capabilities of the underlying computational substrate where services run (e.g. higher volatility, scarcer computational power, or programmability). As a consequence, the nature of the services that can be run on them changes too (smaller codebases, more fragmented state, etc.). These changes bring new requirements for service orchestrators, which need to evolve so as to support new scenarios where a close interaction between service and infrastructure becomes essential to deliver a seamless user experience. Here, we present the challenges brought forward by this new breed of technologies and where current orchestration techniques stand with regards to the new challenges. We also present a set of promising technologies that can help tame this brave new world.

Keywords: NVM, SDN, NFV, orchestration, large scale, serverless, FaaS, churn, edge, fog

1. Introduction

There is a new breed of technologies that are becoming mainstream in current Information Technology (IT) infrastructures. Fog computing aims to partially move services from core cloud data centres into the edge of the network [1]. Thus, edge devices are increasingly becoming an essential part of the IT infrastructure that extends from core cloud data centres to end user devices, allowing some management functions to be offloaded to the vicinity of sensors and other user devices, while heavy analytics can still happen in the cloud, possibly on aggregated data [2]. This is especially relevant for resource-constrained churn-prone devices in the Internet-of-Things (IoT).

The fog has also been propelled by the advent of programmable infrastructures, like Software-Defined Networking (SDN), Network Function Virtualization (NFV) [3, 4], and data centre disaggregation [5, 6, 7]. These have simplified infrastructure configuration for data centre servers, storage, as well as core and edge networks. As a result, the infrastructure is able to adapt to the needs of the services that run on it, making the interplay between the services and the infrastructure more dynamic and complex.

In parallel, recent trends in software such as microservices, foster the utilisation of smaller software functions. Cloud-based serverless computing, also known as Function-as-a-Service (FaaS), is an attempt to tame complexity by dividing services into smaller individual functions that can be deployed and executed independently [8, 9].

These technologies, shown in Table 1, gradually blend together to create a new IT environment characterised by the heterogeneity of equipment, technology and service, large-scale distributed infrastructures, high resource churn, and scarce computational power at the edge. Works that orchestrate serverless functions in an NFV context or amalgamating SDN and NFV orchestration are predominant in the first (left hand size) of the table. The central cell highlights efforts to blend programmable network techniques with fog orchestrators, while the right hand side cell shows works that try to make fog and serverless orchestration converge.

These technologies also affect the nature of services incurring smaller code bases and more fragmented state. The complexity of the resulting IT environment and services makes service orchestration a central task to coordinate and schedule the operation of a myriad of distributed service components. Orchestration becomes even more challenging when different technologies are involved, requiring hybrid solutions that coordinate service provisioning and management taking into account the requirements and the particularities of each technology. While there has been some work [10, 11, 12] on hybrid orchestration of pairs of these technologies, there has been no attempt to com-

Table 1: Summary of papers where these technology trends are converging

Programmable-FaaS	Edge/Fog-Programmable	Edge/Fog-Serverless
[13, 14, 15]	[16, 11, 12, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 10, 27, 28]	[29, 30, 31, 32, 33, 34]

prehensively tackle all of them. The orchestration challenges that result from this hybridisation of technologies are, therefore, still not fully understood.

In this work, we investigate research challenges in next-generation service orchestration frameworks. In particular, we present a comprehensive review with the three following main goals:

1. understand how each of the aforementioned technologies relying on orchestrators change requirements for orchestration
2. reveal challenges for state-of-the-art techniques to meet those requirements
3. discuss potential research directions to tackle new challenges in orchestration systems

The rest of this paper is organised as follows. Section 2 introduces the main technologies where service orchestration is central to provision and instantiate services. Section 3 analyses state-of-the-art orchestration techniques and introduces the main unsolved challenges. Section 4 introduces potential research avenues for these challenges. Finally, a critical discussion of the main lessons of this review work is presented in Section 5.

2. Recent Technological Trends

As can be observed in Figure 1, most of the efforts around orchestration have happened around the cloud [35] and its natural expansion to the edge, via the fog [36]. Classic VM scheduling has partially been combined with edge resource selection and serverless functions in the data centre. As we will see, there are also some efforts on programmable network orchestration and very few dealing with the hardware, which new technologies make highly configurable and subject to orchestration too. In summary, orchestration is central at different levels from the data centre hardware and computing resources orchestration to the network management and service orchestration going from the core to the edge of the network.

2.1. A Comprehensive Motivating Use Case on the New Technology Landscape

A motivating use case of this technology hybridisation process is that of connected cars, which are estimated to produce between 4 and 100 TB of data a day [37]. Their potential need for upload bandwidth poses significant stress in current data and network infrastructures and edge/fog technologies have been postulated as a great starting point to cope with this huge data overload [38, 30].

These technologies can be complemented by smarter network management techniques, where an SDN controller may enable some traffic prioritisation for key data streams to nearby fog nodes (e.g. cars uploading updated information on road conditions to road side "priority event" relays).

At night, benefiting from classic diurnal periodicity patterns in networks, Telco operators use WAN accelerators as NFV functions to speed up the offload of parked car data into car vendor-operated cloud services that may be running in a different continent. In this setting, car-vendor provided specific deduplication and encryption serverless functions can be used to minimise the amount of information to be uploaded in a secure way. In the same vein of data management, [24] proposed an architecture that blends edge, fog and IoT to provide analytic mechanisms for processing (and reducing/aggregating/synthesising) the amount of data that hits cloud servers. [16] proposed a fog node and IoT hub, distributed on the edge of multiple networks to enhance the implementation of several NFV services, like a border router, a cross-proxy, a cache, or a resource directory.

Moreover, for cars driving in smart cities, it may be that CCTV cameras on lampposts may automatically detect blind spots for approaching cars and warn them about people or moving elements that can get into their trajectory. This is a mix of IoT sensors and SDN technologies. Similar safety features requiring local information and real-time processing have been achieved by combining an SDN controller with a Fog controller [25]. The fog offers network

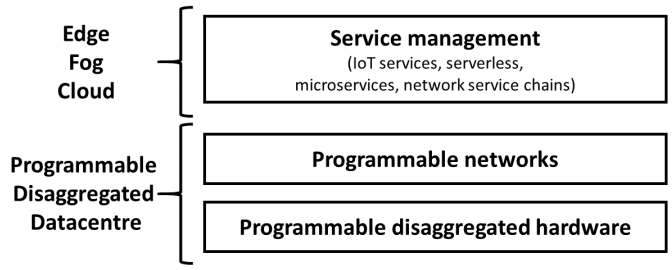

Figure 1: Current State of the Orchestration Landscape. There is a much higher abundance of works dealing with orchestration as we go higher in this stack.

context information, location awareness, and ultra-low latency; which could satisfy the demands of future Vehicular Adhoc Network (VANETs) scenarios.

All of these subsystems are critical to deliver a service where failure, delays, and security issues can be critical. Failing to synchronise the deployment edge nodes running serverless and NFV functions in a timely manner may result in critical security (e.g. unencrypted driver data, hacked cars), safety (malfunctioning SDN-enabled traffic prioritisation for critical events, like revealing moving objects in blind spots), functional (e.g. network clogging due to lack of deduplication or proper WAN acceleration resulting in outdated road maps) issues.

Each of these technologies *per se* presents new challenges for orchestrators, but the combination of (programmable) infrastructure and services creates a novel interplay between infrastructure and service.

In the remainder of this section, we analyse each of currently-deployed technologies separately to extract orchestration requirements. Our ultimate goal is to shed the light on the need for comprehensive orchestration techniques that could coordinate and schedule network services simultaneously through different technologies across the edge-to-cloud network.

2.2. Programmable Disaggregated Infrastructures

2.2.1. Data Centre Hardware Disaggregation

Classic single computer architectures have gradually been split apart, instead giving rise to disaggregated computers where CPU, memory, and storage are interconnected over a high-bandwidth network, rather than over a bus within a single chassis [39, 5].

Such physical separation allows for more flexibility in management and maintenance, catering for different and varying application needs [40]; it also enables more efficient virtualisation of specific data centre resources [41]. Under this model, data centres would not be composed of traditional servers each of them having its own resources (disk, memory, etc.), but they would instead consist of racks of specific resource types that are offered as pools of virtualised resources accessible via the network [42]. The granularity of these resources tends to be more finely controlled as there are strong incentives for cloud data centres to increase utilisation by encouraging use and release utilisation patterns, which require smaller execution units [43].

Several recent efforts follow this disaggregated model of data centre organisation, focusing specifically on flash [6] and traditional storage [44]. While storage disaggregation is common practice (e.g. most cloud vendors offer some volume or elastic block store service), memory disaggregation is another novel trend [7, 45, 46]. Ideas on accessing remote memory were extensively studied 20 years ago and now they are getting revived due to the massive improvement in network latency (3 orders of magnitude). As memory latency has not improved as much, there is a gradual convergence of performance [47].

Several hardware solutions are currently under development both as research projects [39, 48, 49] and industry efforts, such as CCIX [50], Gen-Z [51], OpenCAPI [52], and Omni-Path [53]. However, as data centres expand to the edge of the network, there will be a larger number of edge devices serving as infrastructure. This seems to indicate new protocols to access remote memory with limited (or none) hardware support are needed [47]. Edge devices and cloud will have to adapt to support more abstract configuration and programmability mechanisms.

2.2.2. Programmable Memory and Storage

Disaggregated service infrastructures aim at increasing the utilisation of available resources by separating resources in different pools. This is common in current data centres.

These disaggregated elements are also becoming configurable by software. For instance, storage volumes are dynamically attached to the memory/cpu of a VM when users request it.

Storage systems have improved in speed significantly in recent years [54]. This is largely thanks to low-latency storage (e.g. flash) that moves the bottleneck to be the CPU and network instead of storage where it has traditionally been. Additionally, the rise of *de facto* APIs for network-attached storage systems, such as OpenKinetic key-value stores[1], further contribute to this recent development.

Recent work has focused on developing programmable storage systems that allow the composition of new storage services through the reuse of existing storage interfaces [55]. This is different from the notion of Software Defined Storage (SDS) [56] where storage racks are assembled from commodity hardware.

Memory disaggregation together with the advent of non-volatile memory (NVM) technologies and optical interconnects [57] are also requiring higher memory programmability – see Table 2. In some systems, like HPE's 'The Machine'[2] or FluidMem [46], compute resources can access additional memory in a data centre on demand. In general, the "softwareisation" of the infrastructures enables multi-tenant usage, which will have to be considered by orchestrators when planning resource allocation (see 'multi-vendor / -domain' in Table 2). The possibility of accessing byte addressable memory in nearby edge devices over wireless communications seems less feasible due to excessive latency in current technologies.

As indicated in Figure 1, the number of works dealing with systematic orchestration of disaggregated and programmable hardware is very limited, mainly due to their recent appearance and limited industrial adoption at scale.

Programmable storage systems allow encapsulating storage functionality as reusable building blocks, leveraging storage capabilities through new interfaces. However, the composition of interfaces is complex, and it is important to limit the usage of object interfaces, as is the sandboxing of the runtime space [55].

2.2.3. Programmable Networks

A few years ago, network service management used to be highly manual, resulting in extensive capital and operational costs. A good illustration showing how daunting is to manage network services is the example of provisioning network service chains in traditional networks. A network service chain is an ordered series of network functions (e.g., routers, Firewalls, intrusion detection systems) that process incoming traffic. Traditionally, provisioning such a chain requires a lot of endeavour from IT operators to acquire and deploy networking equipment and to manually configure the network to steer the incoming traffic across the service chain components. These tasks not only require weeks to months to be implemented by also need knowledgeable human resources and a lot of effort. Furthermore, the need for new hardware equipment and human resources incur high capital and operational costs, making this simple service provisioning operation extremely daunting, costly and time-consuming.

The advent of SDN and NFV technologies brought provisioning time from the scale of weeks and months to that of minutes and significantly reduced human intervention by automating all the service chain provisioning steps. On one hand, SDN has succeeded to revolutionise the way network components are configured and managed. In traditional networks, each networking equipment consists of: (i) a control plane, that is mainly responsible for taking routing decisions; and (ii) a data plane, that is in charge of forwarding traffic according to the decisions made by the control plane. SDN technology moves the control plane from the network equipment to a logically-centralised software-based controller, making it possible to manage the whole network from a single point of control. This offers network operators the programmability and the flexibility, allowing to easily configure their networks and dynamically adapt their routing paths to applications' performance requirements. In addition, SDN provides the tools to set up bespoke control over the traffic, allowing to define fine-grained flow forwarding rules (e.g., flow definitions are easily performed by the network operator using diverse information in packet header fields).

[1] https://www.openkinetic.org/

[2] https://github.com/FabricAttachedMemory/tm-librarian

On the other hand, NFV is a new technology that leverages server virtualisation technology to turn network functions (e.g., routers, firewalls, proxies) that traditionally used dedicated hardware (middleboxes or network appliances) into software that run on top of general purpose hardware such as virtual machines (VMs) [4]. NFV inherits all the advantages of virtualisation. For instance, it offers the possibility to create different types of network functions whenever needed and to adjust their processing capacity to the varying demand. It also allows to easily create a copy of a network function or simply migrate it to another location if and when required.

The combination of SDN and NFV technologies makes it now possible to provision within very short timescales a fully-fledged network service chain. In this context, service chain provisioning and orchestration is one of the most challenging problem as it requires solving several optimisation problem simultaneously. Challenges include placing virtual machines, connecting them and steering the traffic through the ordered chain of the network functions. This is a non-trivial task as it requires finding the best compromise between VMs' hosting costs, bandwidth costs, and performance requirements in terms of the chain end-to-end delays [58].

The orchestration framework needs also to manage other features like multi-tenancy, fault-tolerance and availability management. Multi-tenancy allows multiple users to share the same infrastructure and hence requires resource isolation between different service chains and better performance management to satisfy each tenant's requirements. The orchestrator needs also to ensure high service chain availability through efficient fault-management. This requires leveraging SDN and NFV technologies to put forward a set of solutions allowing to handle different types of failures (e.g., node, link and software failures) and mitigate their impact on service chain availability.

2.3. Edge and Fog Computing

Edge and fog technologies shift centralised cloud computations toward edge devices in order to decrease latency, improve spectral efficiency, enable enhanced context-specific functionality, and support the massive machine-to-machine type of communication. They also allow for localised functions such as processing that benefit from p2p-style communications and exploiting nearby resources [59], exploiting data locality to obtain faster results.

This is driven by the explosion in the number of connected devices and services, and the increasing demands of applications for low latency and interactive experiences [32, 36].

Additionally, fog computing is facilitated by the availability of suitable hardware in the form of small, affordable, low-power computers [30], along with improvements in virtualisation technologies that enable slicing resources between different users to provide isolated environments. This contributes to a need to support highly heterogeneous hardware and software stacks that emanates from fog/edge environments and is improved thanks to the usage of programmable infrastructures (Table 2).

There is a need for a close interplay between cloud and network resources, where resources can be dynamically instantiated. For example, [60] describes how to dynamically deploy cloud storage services in core networks to simplify data backups across data centres. This example requires: (i) Partial view of the capabilities of other ISPs (ability to select services and providers); (ii) Dynamically instantiate/select virtual resources (e.g. virtual routers and storage VMs); (iii) Ability to guarantee predetermined QoS levels across all the services composed to deliver a network function [29]; (iv) Monitor quality metrics and automatically re-configure the VNF, if needed.

Similar techniques are being deployed at the edge. The main cloud vendors (such as Amazon and Microsoft) are making it easy to integrate IoT devices into their clouds mainly by providing gateway management and tight integration with their other cloud services [61, 62, 63, 64].

While there are interesting works coping with resource provisioning, deployment, and scaling of applications on edge devices [65, 66], there is still a gap that needs to be filled before accomplishing the vision of a more decentralised infrastructure where the devices can synchronise themselves without the need of central cloud services [1]. This is where the usage of serverless functions may come in handy, since they can be run at the edge on locally generated data, and rely on the cloud for management, analytics, and durable storage [61]. The orchestration of these small functions that run scattered across the edge of the network and the interaction with centralised cloud services is still a challenge [32] (larger-scale and finer-grain required – see Table 2).

While serverless frameworks provide facilities to define dataflow orchestration [67, 68], they generally ignore how to enable inter device communication. For instance, a set of devices may choose another, more powerful, neighbouring machine to do some aggregation before sending the data to the cloud. Also, current orchestration approaches do not

cope well with device churn (another essential requirement for orchestrators – see Table 2), shadow devices being one of the few solutions out there [61].

[69] defined fog cells, as single IoT devices coordinating a group of other IoT devices and providing virtualised resources. These resources are located close to the edge of the network next to the data sources or sinks, instead of involving the cloud. They build on the idea of fog colonies (also referred to as edge clouds by [1]), which are micro data centres made up from an arbitrary number of fog cells. Fog colonies distribute task requests and data between individual cells, allowing for cloud offloading and multi-cloud deployment [70].

We envision colonies and cells as dynamic entities that can be formed based on short term convenience: for instance, summer campers establishing a mesh network with nearby devices to enable local connectivity in areas with poor service (e.g. each phone relays nearby messages) [32]. We refer to these dynamic and transient cells and colonies as fluid distributed organisations. A fog node can start autonomously, and become the lead orchestrator or part of a set of distributed orchestrators reaching some quorum before making decisions and dynamically leave the colony without warning [71].

In this setting, hierarchical approaches tend to fall quite short; their scalability and manageability become harder as scale, distribution, and churn increase [72]. Also, most prior approaches tend to assume that a device participates in a single cell or colony, while in reality it may belong to several of them at the same time. Moreover, multi-tenancy is a very uncharted land (see Table 2). As edge devices access the network via different ISPs and potentially operate across various administrative domains [73], the ability to deal with multi-organisation/multi-tenant environments becomes a must for any orchestration technology.

Edge/fog devices run on energy-efficient low spec'ed hardware running smaller execution units. The basic units of execution have continuously shrank: from VMs to containers, microkernels, and serverless functions. Such smaller execution units are dictated by the need for increased flexibility and control, and also the strong economic incentives to increase resource utilisation under usage variability [43]. As a consequence, the number of lines of code associated with each unit of execution, the duration of each execution unit, and the amount of state it keeps have all greatly decreased. More devices (larger scale) come together with finer grained units of execution (data and code) (see Table 2).

Radio access, network transport, and cloud resources are coordinated by a higher level orchestration layer [20, 21, 22, 27]. These three different resource types are a simplification of a much more complex mixture of networking technologies and protocols [74]. Fog and edge computing create a scenario of unbounded heterogeneity, or *hyper-heterogeneity* as it affects devices, software stack, communications, and data management, and hardware technologies.

Orchestrating the security in Edge-Fog computing and IoT is a huge challenge. It encompasses different technologies from different fields, including, among others, wireless-cellular networks, distributed systems, virtualisation, and platform management [75]. It imposes new models of interaction among different heterogeneous clouds, which require mobility handover and migration of services at both, local and global scale. The threats from all those building blocks are inherited in the edge-fog paradigm. Different layers are affected by diverse technologies that need to interoperate to achieve holistic and confidential connectivity [76].

At the edge, services and devices can be compromised through different attacks such as privilege escalation, service manipulation or even physical damages in unprotected edge/fog/IoT environments. At the network level, edge-fog and IoT impose additional threats, as wireless connections might provoke man-in-the middle attacks [77], spoofing attacks, eavesdropping or traffic injection, or authentication [78] to name a few. Privacy leakage is also an important threat at the edge, as mobile devices can be tracked without user awareness. In this sense, Roman et al. [79] have recently identified main security threats and challenges in mobile edge and fog computing.

Aspects such as certification of virtualised applications, tenants data isolation and sharing, resource usage control still require the definition of edge device policies and specific access control mechanisms [80, 81]. It is hard to establish a chain of trust with edge devices under different organisational boundaries. Dynamic and multidimensional trust and reputation access controls mechanisms have been suggested [82]. Also, privacy-preserving mechanisms, satisfying "unlinkability" and minimal disclosure properties [81], as well as data aggregation schemes to conceal user data, are needed in computing-enhanced IoT applications [83].

Expressive languages for defining high level security policies and models could serve as input for orchestrators to organise and choreograph the aforementioned security services.

Table 2: Emerging Orchestration Needs in New Technologies

Technology	Functional Orchestration Needs	Requirement
NFV	quick routing adaptation [20, 84, 85]	Dynamism and Speed
	heterogeneity in infrastructure and VNFs [20]	Heterogeneity
	reduced human involvement [20]	Automation
	comprehensive NF, lifecycle management; virtual tenants [84, 86, 87]	Multi-tenant
	coordination across orchestrators [85, 86], move to edge [85]	Multi-domain
	single function scheduling [86, 31]	Finer grain
	function splitting [86, 31], deployment & config checks [22]	Larger scale
	Virtualisation issues [88], security scalability, [89] availability [90], vulnerable applications [91], topology validation [90]	Security
SDN	controller planning [92, 86, 87], move to edge [19]	Multi-domain
	heterogeneous network [87], end-to-end connectivity [93], move to edge [19]	Heterogeneity
	end-to-end connectivity [93], move to edge [19], global path computation [17]	Larger scale
	east-west confidentiality [94], availability [95]	Security
	blend SDN/NFV orchestration [86, 87]	Multi-technology
Programmable storage/memory	composition of storage service though separate storage modules [55]	Scalability, Finer-grain
	remote/disaggregated (edge) memory [57, 46, 47]	Multi-vendor, Multi-domain
	protection of new programmable storage interfaces, sandboxing [55]	Security
Fog/Edge computing	dynamic coalitions of edge devices and cloudlets [69, 1, 71, 32, 96, 97, 59, 98, 36]	Dynamism, Churn, Scalability, Locality-awareness
	going beyond shadow devices for reliability [61, 71, 32, 96]	Churn
	dynamic end-to-end service availability [60, 71]	Multi-tenant, Multi-domain
	smaller execution units [43, 32, 97, 66]	Larger scale, Finer grain
	diversity [21, 22, 27, 74]	Heterogeneity
	M2M confidentiality, wireless-based attacks [79], trust management [82] AAA [78] [77], privacy-leakage [83].	Security privacy
	ensure quality-of-service on a variety of infrastructure elements [29, 36]	Heterogeneity, Multi-domain
Serverless computing	reduce latency in function execution and state handling [99, 9]	Speed, Locality-awareness
	going beyond shadow devices for reliability [61]	Churn
	FaaS and Serverless security issues [100]	Security
	smaller execution units, smaller state [8, 101]	Larger-scale, Finer-grain

2.4. Serverless Computing

For over a decade, large-scale complex computations have shifted to a high-level, function-oriented model in which computation is expressed as functions that are composed into dataflows, and automatically deployed and managed by a cluster [102, 103]. From the point of view of a developer, this level of abstraction splits the jobs that are to be executed from the way they are provisioned. Resources are automatically freed when computing jobs are done.

Similarly, the concept of "serverless computing" (FaaS) refers to server-side logic written by the application developer, running in fully managed stateless compute containers that are event-triggered, and ephemeral (may only last for one invocation).

Fundamentally, the serverless paradigm completely decouples running code from the management of the supporting server applications. This is a key difference compared to other modern architectural trends like containers and Platform as a Service (PaaS). PaaS/Containers applications are not geared towards bringing entire applications up and down for every request, whereas FaaS platforms do exactly this.

From the provider's perspective, many cloud deployments tend to be very static as users deploy VMs for long periods of time. In contrast, the variation in memory and CPU utilisation tends to be much more variable [43], and thus offers a better way to optimise resource usage and to bill users. Therefore, cloud vendors have strong incentives for services to be built on serverless architectures as opposed to following fixed-price model for long-running VMs [8]. The billing model is based on the number of function invocations and how many GB-s the function uses. Thus, developers have strong incentives to build smaller functions that minimise execution time and memory consumption. This is a factor contributing to the 'finer-grain/large-scale' requirement in Table 2.

Function initialisation and state recovery/storage are key elements that may slow down function execution, resulting in potentially more expensive executions or totally failed function chains [99]. These contribute to the "Speed" and "locality awareness" requirements of next generation orchestrators (see Table 2).

Delivering cloud applications typically means composing different tasks (hence the need to orchestrate function composition that comes as a requirement in Table 2). One way to deliver cloud applications in a serverless way would be creating a function for each task. However, orchestrating those small functions (finer granularity than a microservice) could be really hard to debug and optimise [101].

Initialisation latency, state push/pull, and availability problems become significantly worse as we move to an edge/fog computing arena [8, 99, 9]. Being able to control co-location of fine-grained code with tiny subsets of the data may prove essential to deliver appropriate orchestration capabilities in serverless environments (see Table 2). The ubiquity of edge devices and the advent of fog computing, together with the finer-grained nature of typical serverless functions, have changed the options available for orchestrators to play a key role.

FaaS moves some of the security concerns from the user to the platform provider. As it is remarked in [100], users do not need to take care of OS patches anymore, but security updates to 3rd party dependencies of applications remains the same. In FaaS servers are immutable and short lived, minimising the possibility of a long lived compromised server. However, security monitoring and accounting and debugging becomes more difficult [9], as traditional monitoring agents need to be exposed by FaaS providers.

2.5. Current Standardisation Efforts

The term orchestration is used pervasively in the literature reviewed so far. The end result is a myriad of often incompatible standards that tend to cover one of the new technological trends. In this subsection we present the most prominent approaches related to these trends.

The ETSI NFV Management and Orchestration (MANO) is one of the most solid pieces of work in terms of NFV standardisation [104]. [71] have suggested that NFV should be the starting point for new standardisation efforts beyond NFV, given the need to define, for each application domain, the scope, properties, and requirements of service orchestration concepts is not different in the Fog Computing environment. This suggestion has also been followed by project Tacker in an attempt to integrate a MANO-compliant orchestrator on top of one of the most widely-used cloud management suites, OpenStack[3]. ETSI MANO deals with computing nodes where only CPU, memory, network,

[3] https://wiki.openstack.org/wiki/Tacker

hypervisor, and Operating System can be chosen. In the edge/fog, devices are much more heterogeneous and have capabilities other than that(e.g. sensors and actuators). These unreliable devices play a major role in the process and need to be taken into consideration in the architecture or an orchestrator.

The ETSI Mobile Edge Computing (MEC) Reference Architecture [105] emphasises the need to consider a comprehensive set of constraints, and also refers to triggered application instantiation and relocations, one of the characteristics of our solution, similar to a fog orchestrator [71]. However, a detailed specification or/and Reference Architecture for the orchestrator is still missing.

The Open Networking Foundation (ONF) has been working on SDN standardisation for quite some time. Most of their documents treat orchestration as an external (client-driven) coordinator of several SDN controllers [106]. This is similar to the recently released OpenFog Reference Architecture [38], which mainly focuses on the fog node.

The Topology and Orchestration Specification for Cloud Applications (TOSCA) is turning into the *de facto* standard for modelling service orchestration [107]. TOSCA is especially suited for defining services, their building blocks, requirements and capabilities, but it still does not help solve problems like device churn, multi-domain/multi-tenanted orchestration of tiny functions at scale. In addition to TOSCA, there are a few others like the IETF NETCONF Data Modelling Language (NETMOD) WG, together with recent expansions for VNF network services (following the ETSI MANO architecture). More acronyms would be needed for a top-down end-to-end solution (e.g. cloud VM configuration languages, network traffic engineering configurations, etc.).

Current trends for service description and modelling are very prescriptive and fragmented. Recent standardisation efforts for the new technologies presented in this section are poorly specified and cannot cope with all the new requirements.

3. Current Orchestration Challenges

Building on the previous section, here we analyse the requirements in Table 2 in depth, trying to systematically unveil orchestration challenges and attempts to tackle them.

3.1. Churn and Unreliability

Edge resources are inherently volatile. The advent of the fog extends the cloud to the edge to a point where end user devices could be employed as infrastructure to deploy services or service functions [59]. Moreover, as the fog is increasingly being used to support transient FaaS, e.g. to support context-specific mobility functions, functions are ephemeral which imposes a rate of change much higher than in cloud environments.

Such nature poses significant challenges on different functions that enable orchestration. Description of resources and functionality might not always be accurate as it could quickly be outdated, which complicates reliable deployment and service level agreement (SLA) guarantees. Similarly, discovery must be dynamic in order to take advantage of new resources as they become available, as well as move away from decommissioned/failed resources (see Table 3). Further, monitoring needs to always seek up-to-date information, avoiding obsolete data and empirically complementing self declaration from devices.

Discovering mobile edge devices by scanning all connected communication interfaces and enlisting all locally available mobile edge devices is a first common approach [24]. An example of this prevalent approach is Foggy [23], that relies on a centralised orchestration server and a container registry for deployment. However, high churn may advise against this practice since it may exhaust remote device batteries or increase the energy bill of the infrastructure provider(s).

In contrast to the highly stable resource provisioning in cloud data centres, edge devices can be switched off dynamically, in order to cater to edge workloads and latency-, location- and privacy-specific requirements. As such, the edge is a highly volatile operational environment where resource availability is liable to significant changes over time and is divided across multiple domains of those operating the edge resources. The fog acts as a stabilisation layer, offering more reliable infrastructure in the proximity of the edge devices. Still, dealing with orchestration in unreliable environments comes with specific challenges to each of the phases of the orchestration process.

3.2. Heterogeneity

Inherent in the task of orchestration is the dealing with resources of varying nature and access methods, and that are managed under different administrative domains. Furthermore, the fog paradigm offers an alternative to the centralised model of the cloud. As such, any attempt to resolve the above challenges through central elements to handle monitoring, scheduling, configuration, etc. would undermine the benefits of disaggregation [67, 71].

These challenges call for two main approaches to orchestration. First, distributed orchestration is essential to deliver the potential of the fog paradigm, where orchestration elements manage different edge domains and inter-coordinate in a hierarchical or peer-to-peer fashion. Currently available tools, such as IBM Node-RED[4], offer high-level developer tools for creating interconnected flows. However, they are tailored specifically towards IoT functions. More generic tools are needed in order to support rich and customised coordination between a distributed network of orchestrators. Second, a sophisticated level of abstraction is needed to hide away the complexity of heterogeneity from application development and deployment processes. Toolsets are needed to not just simplify the tasks of resource discovery and monitoring, but also end-to-end lifecycle management, and to compose elaborate adaptive migration policies and mechanisms.

Managing heterogeneous resources across distinct administrative domains is already a challenge, but the independence between resource management and workload scheduling on fog devices increases the difficulty of their orchestration.

3.3. Dynamism

The IoT brings *ad-hoc* devices at the edge of the network as infrastructure to run services. The quickly changing network conditions at the edge bring a significant amount of additional dynamism to service-based applications, in contrast with the relative stability of large data centres. Dynamic adaptation mechanisms, including runtime configuration, deployment, switch-over will be vital to be orchestrated across the infrastructure.

Dynamic deployment is achieved by integrating continuous deployment technology on the edge of the network while coping with IoT's intrinsic heterogeneity [23, 66]. As IoT devices may be offline for quite some time, an orchestrator needs to gracefully cope with loss of connectivity [89] and increased likelihood of failing devices [32].

Meeting the need of dynamic reconfiguration at the network level can increase network incidents and temporary malfunctions. A service orchestrator will also have to provide network diagnosis and root cause analysis during service disruptions [89]. In parallel, the orchestrator must support network resource scheduling that can adapt to near real-time service demands [108]. [96] focus on reliability of orchestration for IoT domains, proposing autonomous mechanisms that enable the "analysis and management of: (i) the overall system goals describing the required applications, (ii) the composition and requirements of applications, and (iii) the constraints governing the deployment and (re)configuration of applications".

Service oriented orchestrators for network functions have been proposed [20, 29]. New techniques aim at integrating application information into orchestration decisions [109]. Responding and adapting to specific application needs, while optimising resource usage can be an impossible mission and has been tried before with little success in practice.

IoT services can often be choreographed through workflow or task graphs to assemble different IoT applications [110, 98]. In some domains, the orchestration is supplied with a plethora of candidate devices with different geographical locations and attributes. In some cases, orchestration would typically be considered too computationally intensive, as it is extremely time-consuming to perform operations including pre-filtering, candidate selection, and combination calculation while considering all specified constraints and objectives. Static models and methods become viable when the application workload and parallel tasks are known at design time. In contrast, in the presence of variations and disturbances, orchestration methods typically rely on incremental scheduling at runtime (rather than straightforward complete recalculation by rerunning static methods) to decrease unnecessary computation and minimise schedule makespan [32].

[111] propose a semantically enhanced mechanism to define quality-of-service for web services (see Table 3). Similar techniques are likely to become more pervasive, but creating, adapting, and adhering to fixed ontologies has not proven highly effective.

[4] `https://nodered.org/`

There are operational needs about the speed at which the orchestrator solver can process incoming monitoring information and return a fast and accurate enough decision. [97] focus on softening the orchestration decision making process to cope with scale. [96] to formulate Satisfiability Modulo Theories (SMT) constraints that define desired system properties, enabling the use of SMT solvers to adaptively compute optimal system (re)configuration at runtime (see Table 3).

3.4. Large(r)-scale and Fine(r) Grain

As more devices are connected to edge networks and fog environments and the size of the unit of execution decreases (as described above) [36], it will be more difficult for cloud orchestrators to make a decision before the information they rely on becomes obsolete.

Service description must support aggregation/abstraction of resources in some ways to help with scalability (e.g. using hierarchical models), but the descriptions will have to be more abstract to cope with more devices and finer-grained execution units (with smaller state) – see Table 3. [26] offer a high level declarative language to describe implementation heterogeneous devices. Abstracting masses of IoT devices with heterogeneous capabilities remains a hard problem.

The scale that the fog/IoT impose on next generation orchestrators calls for mechanisms to describe the way data is handled, as the interplay between the swarm of devices executing the application and the data becomes more critical to achieve the required performance goals (see Table 3). [26] suggested a design-driven approach that can be leveraged in two ways: first, design declarations are used by a compiler to generate a customised programming framework. These declarations can be supplemented with information to expose parallelism and allow efficient processing of large data sets.

Fog colonies/edge clouds may be distributed across a rather large area, interconnected through heterogeneous networks, while cloud resources are usually placed in centralised data centres. Discovering, selecting, and deploying devices can be built in 3 different ways: 1) hierarchical name system (like the Domain Name Service); 2) in an unstructured P2P flooding fashion ("ask your neighbour"); 3) hybrid (P2P at the edge), hierarchical there after (relying on nearby cloudlets [112] and using central cloud services as a last resort).

Declarative model-based languages have been there for a long while and they are used by developers to express their resource needs and define preferred configurations in a more generic manner, rather than specifying the individual configuration of millions of devices [113, 114]. Mapping from these high-level configuration languages into finer-grained tiny units of execution is an open research challenge, but some solutions are already under way: [26] defined DiaSpec, a declarative language to describe the functionality of an IoT device, abstracting over the specific hardware and implementation. These declarations consist of source and action facets depending of the functionalities to be described. Each device of that type needs to conform to the interface and implement the sources and action operations. They also define a set of higher level constructs to work with large masses of sensors.

Since resources are disaggregated, there is more flexibility and less cross-configuration interactions (e.g. a networking configuration affecting storage read/write throughput) but orchestrators need to become more robust.

Configuration consistency also becomes an issue. An example would be setting up a high-speed channel between two functions executed in two different continents by orchestrating serverless environments as well as network control plane (say some traffic engineering is needed) and modifying VNFs along the data path (e.g. opening firewalls transiently). If the changes are not properly orchestrated, a sending function may send data through a data path that may filter or slow down that type of traffic. Synchronised clocks can be used to reduce the probability of having a violation of external consistency [115], but atomic clock synchronisation may be required for extremely latency sensitive orchestrations [116].

[33] define virtual fog functions (VFF) and several strategies to deal with mismatches between VFF and the capabilities of the underlying hardware in the IoT devices. This work suggests that a more interactive generation of orchestrators is needed, where the developer is in the loop at least at deployment/configuration time.

Osmotic computing considers computational infrastructure as a chemical solution whose properties can change over time, the focus is on identifying the properties of what constitutes a solute and solvent, which is then operated on the principle of osmosis to manage and control services [18] (see Table 3). Identifying how microservices can be migrated from edge resources to cloud-based resources (and vice versa), and what are characteristics influencing such migration, remains a challenge. The right formats and protocols for this to happen and cope with churn are not yet clear [117].

3.5. Speed

FaaS allows fine-grained, highly dynamic configuration. FaaS divides microservices in smaller software chunks that can be executed very quickly (price based on cpu/mem usage is a strong incentive to optimise function execution).

Smaller execution units that complete in seconds are a better fit for short lived resources, where failure is commonplace. This is a challenge for orchestrators, that need to decide where to execute a given FaaS function and reschedule (or take preemptive executions) to cope with failure. Slow, batch-style global optimisation is no longer an option. Instead, online-style techniques, with deadlines need to be explored in order to take advantage of the flexibility of functions.

A main feature of serverless computing architectures is the ability/need to deploy new instances in the time scale of ms [8]. This is also true for supporting flash events where millions of customers hit a website for a specific sales promotion, for instance. NFV functions need to be deployed/undeployed in sub-second time periods.

Containers are the basic unit of deployment for serverless and many NFV functions. Container-based FaaS services tend to reuse the same container to execute multiple functions, even with this optimisation, serverless functions are significantly slower than containers at low request volumes [8]. Some tricks to avoid the overhead of using persistent block stores to fetch data and configuration are possible. A scheduler aware that two different functions rely heavily on the same packages can make better placement decisions.

Session locality is an important factor (see Table 3): if a function invocation is part of a long-running session with open TCP connections, the orchestrator should run it on the machine where the TCP connections are maintained (avoiding traffic diversion by proxies). Mobility management however brings additional challenges to the orchestrator.

Also, data locality will be important for running serverless functions pulling/pushing or scanning through massive state (see Table 3). Orchestrators may require prediction capabilities to anticipate what data a particular function will read, making sure it is available to be function on time.

All these problems with data and code locality exist at a data centre level, but they become more critical at the edge of the network. Advanced techniques to determine which edge nodes should be used to share the workload with and how much of the workload should be shared to each node are needed, heterogeneity and churn being the main deployment challenges [14].

3.6. Chaining Heterogeneous Functions and Storage

IoT infrastructures are often modelled as a dynamic graph [32]. IoT configurations can be seen as a graph where the nodes represent the configuration and the edges the dependencies between task. Graphs are also used to describe virtual functions in NFV environments (see the NFV Management and Orchestration spec) and some serverless environments too (e.g. Oracle's Flow Fn serverless orchestration). This feels like a very intuitive approach, but some other serverless vendors orchestrate functions using pre-defined state machines (e.g. [118, 101]).

In principle a similar approach can be used to declaratively describe the high-level features of function compositions in a fog environment, delegating lower level details to the orchestrator. The orchestrator would need to map these high-level descriptions to vendor specific implementations (e.g. 'key-value store' would map to different AWS or Google Cloud products). Locality and heterogeneity however bring additional challenges to the orchestrator.

The standard model of cloud data stores abstracts the physical location where information is stored. However, in a fog environment, together with the mentioned architectures that disaggregate storage from computation, information will be more fragmented than before, and the actual location where these (potentially small) data elements reside can be critical. Hence, location information needs to be included into the high-level descriptions used by orchestrators.

Blending out together FaaS and NFV functions and their interactions with storage can be difficult (e.g. serverless does not support the same conventions that MANO NFV does). An orchestrator will need to not only have a compatible high-level description for all these elements, but also will need to have the right network access rights for interacting with each sub element of the system.

3.7. Fine-grained Locality

As the way to develop applications changes towards microservices, and FaaS, these concepts bring additional questions to how to discover existing functionality.

The discovery of *ad-hoc* services and available computing resources in the fog/edge needs to go beyond beyond predefined contracts and addresses. The probability of trying to contact a device that is no longer available is much

higher at the edge, making device/service registries very ineffective. Mechanisms to initiate local p2p-style resource discovery at the edge have been suggested [32], getting inspiration from the world of *ad-hoc* networks.

Addressing end point mobility with session continuity is another solution to the problem of naming an churn [119]. The Locator Identifier Separation Protocol (LISP) allows an endpoint to switch between networks while keeping its Identifier (IP address) intact by maintaining the Routing Location (RLOC) of each Identifier in a mapping system, which is updated by its control plane. Also, Multi-Path TCP (MPTCP) defines TCP sub-flows at the transport layer based on the IP addresses of all the enabled interfaces on a device. Under mobility, whether the device changes its IP or switches radio technologies (e.g., WiFi to 4G), the new IP address is registered and a new sub-flow is opened. This strategy allows for seamless mobility of the device across networks and radio technologies (see Table 3).

The selection of the most appropriate function to be used is going to depend on where it needs to be executed. For instance, mobile services change physical locations and may require resources *en route*. Thus, more expressive description mechanisms are needed to define these situations (e.g. hardware dependencies).

As for deployment and configuration, edge devices also need to be able to self-manage with little coordination from a central cloud location. [120] have recently developed a system based on edge communications with minimal cloud-driven coordination. The authors rely on edge (locally cached) content as clustering classifier, where a local leader coordinates communications for data retrieval and update. In a fog environment, a coordinator can be in the nearby fog layer, so as to cope with edge device churn.

As mentioned above, deploying lightweight monitoring modules that interact with the orchestrator, but do not overburden the edge devices, seems essential to reach a fair balance between synchronisation and network load.

3.8. Multi -organisation/-tenant Orchestration

Next generation clouds have to orchestrate resources from multiple administrative domains, this can be seen as an extension to the edge/fog and volunteer computing paradigms. While there are companies providing single domain facilities, the need to cross administrative orchestration has become much more pressing to take advantage of these latest trends. Cloud standards have failed to gain traction, but the need to find mechanisms for bridging the heterogeneity gap between platforms, and enabling data integration are more relevant than ever.

Most orchestration technologies working across administrative domains use a broker to orchestrate resources at different levels within a provider (e.g. the cloud and the edge network) and across providers (see [121] for a recent example). Broker models across providers and multi-stage schedulers and optimisers have been quite common in distributed computing and networking since at least 20 years ago [122].

Brokers have also been recently suggested as a viable model for cloud orchestration [123, 124]. As the number of cloud vendors is limited, it is possible to build adapter and brokering layers that tried to homogenise access to different clouds. However, the hyper-heterogeneity and massive scale of edge/fog deployments makes this approach unfeasible.

Network functions can be dynamically discovered, negotiated and elastically composed as services, application service providers may lease VNF chains with given communication capabilities from different ISPs and compose them to operate an end-to-end virtual service infrastructure to offer value-added application services to users (e.g., delay-optimised infrastructure for high-definition video applications [125].

One open question in most academic works is how to handle multi-tenancy and how to scale identity management services to a global scale [81] (see Table 3). Another often overlooked aspect is that at any point in time, some devices may belong to multiple organisations at once (not all users from the same organisation).

3.9. Security and Privacy

SDN-based IoT and Fog networks are vulnerable to the new-flow attacks, which can disable the SDN-based IoT by exhausting the switches or the controller. In this sense, [136] authors present a smart security mechanism (SSM) to defend against New-Flow Attack in SDN-Based IoT differentiating new-flow attack from the normal flow burst by checking the hit rate of the flow entries.

Regarding security in SDStorage, in [131], a software defined based secure storage framework is proposed. Every storage control and security mechanisms are abstracted out from the hardware devices in the data plane and set inside the controller, enabling a centralised decision point based on security policies. Thus, when a host sends storage control

Table 3: Mapping of how recent research contributions in the area of orchestration can contribute to the requirements identified above (in brackets). * means it applies to all requirements above.

Recent Accomplishment
Expressive declarative descriptions **[heterogeneity, dynamism, churn, larger-scale, finer-grain]** [113, 92, 26]
State-machine based function orchestration definitions **[dynamism, churn, larger-scale, finer-grain]** [101]
Dataflow-based function composition **[*]** [99, 68]
Decoupling name from resource (LISP/ROC/MPTCP) **[*]** [99, 68]
Semantic quality of service **[dynamism, larger-scale]** [111]
Spliting services into finer grained functions ("FaaSification") **[finer-grain]** [31, 126]
Data-aware config **[*]** [31, 24]
Trust-management, AAA, Channel-Protection **[Security]** [81, 127, 128, 129]
Serverless/FaaS isolation **[Security]** [130]
Security coordination (e.g. AAA, Trust) in Software Defined Storage **[Security]** [131]
Fluid (edge-fog-cloud) resource allocation/coordination **[*]** [18, 4, 86, 87, 32, 119, 121, 25, 16, 24, 11, 12]
Working under different communication models (edge - p2p; fog - hierarchical; cloud - centralised) **[dynamism, churn, larger-scale, multi-domain]** [32, 24]
State (device and service) prediction **[dynamism, churn, larger-scale]** [32, 14]
Softened goals in service/function composition/configuration **[larger-scale]** [113, 58, 97, 132, 32]
Fluid (edge-fog-cloud) resource allocation/coordination **[*]** [18, 4, 86, 87, 32, 119, 121, 25, 16, 24, 11, 12, 133]
Failure-tolerant orchestrator, cope with stragglers **[dynamism, churn, larger-scale, muti-domain]** [71]
Brokered (multi-domain) hierarchical orchestration [123, 20, 21, 22, 27, 121]
Service support in orchestration decisions**[*]** [109]
Developer support in orchestration decisions("device in the loop") **[*]** [33, 26]
Self-protection, self-healing, self-repair, DoS protection **[Security]** [134, 135, 136, 137]
Universal identity management **[dynamism, churn, larger-scale, multi-tenancy, privacy]** [81]

packets and data traffic to another host in the network, the security controls such as authentication and filtering take place at the control layer instead of at the device level.

Regarding serverless and FaaS security, some initial works are starting to provide isolation at microservices and Serverless computing. Recently, Bila et al. [137] propose a policy-based improvement in the serverless architecture to guarantying and rebuilding vulnerable containers, that can be included as part of the security orchestration. Containers might have built-in vulnerabilities because of wrong configurations or just by the fact of including executable binaries with security flaws. In [130], authors propose a Security-as-a-Service approach for microservices-based cloud applications, providing a flexible monitoring and policy enforcement infrastructure for network traffic to secure cloud applications. Unfortunately, there exist still few research and solutions aimed to cope with the emerging security issues in that field.

With regard to Edge and Fog computing, [127] identified authentication at different levels of the gateways as the main security issue in fog computing. Multicast authentication [129] or decoy information technology technique [138] have also been proposed to withstand malicious insiders by disguising information to prevent attackers from identifying customer's real sensitive data.

In [81] authors explore the idea of privacy-preserving global identities that are universally valid for an entity. Such a feature is essential for handling authorisation in a large-scale distributed environment.

[139] authors present a lightweight privacy-preserving data aggregation scheme, for fog computing-enhanced IoT that can aggregate hybrid IoT devices' data into one in some real IoT applications, so that user private data is concealed.

Recently, authors in [140] identified main attacks that can occur at the Edge-Fog and IoT, including, among others: DDoS, Routing attack, Sink node attack, Direction misleading attack, Black hole attack, Flooding attack, Sybil attack or Spoofing attack. To cope with those attacks the main countermeasures at network layer, focus, nowadays, on ensuring confidentiality, integrity and availability. To this aim, novel end-to-end encryption mechanism specially devised for IoT at different levels (e.g. 6LowPANs encryption, IPsec tunnels, DTLs), including Peer to Peer authentication and key negotiation management, can be orchestrated and configured on demand at the edge as security VNFs.

Trust-management in distributed scenarios such as inter-clouds have been addressed recently [141], where a semantic-web approach is followed to quantify dynamically trustworthiness and reputation among different clouds and services in order to establish reliable federations and communications among the parties.

Malicious and curious adversaries (e.g. MEC data centres) can represent a privacy threat to the Edge/Fog users as they can gain some user-related information in the decentralised ecosystem.

In addition, new cybersecurity orchestration will need to provide self-protection, self-healing and self-repair capabilities through novel enablers and components [135] [134] at the Edge. To achieve those properties, services running in this environment need to work together with a lightweight (potentially distributed) watchdog that sends events to the orchestrator (e.g. compromised device triggers removal of keys and migration of data), to make reconfiguration decisions accordingly.

3.10. Grouping Challenges

The text in bold in Table 3 shows a list of requirements that will be needed by next generation orchestrators. Most of these requirements have been tested in isolation. A more comprehensive orchestration approach joining all of them (and a few others that we highlight in the next section) would still be needed.

This conclusion arises from how orchestration has evolved in the last 20 years: orchestration challenges have evolved over time but mostly in separated areas that are now increasingly more unified due to the "softwareisation" of IT. The orchestration needs can thus be classified in several waves:

1. 1^{st} **wave**: software placement and communication in distributed (sometimes across domains) environments.
2. 2^{nd} **wave**: same as 1^{st} wave but including edge and for resources in the IoT together with programmable networks and serverless functions.
3. 3^{rd} **wave**: same as 2^{nd} wave but adding hardware programmability and disaggregation also at the edge and simplified data management (e.g. [142]).
4. 4^{th} **wave**: same as 3^{rd} wave but with abstracting the underlying heterogeneity, complexity and dynamism of the IT infrastructure making it easier for human administrators and developers to use.

Taking this classification and Table 3 into account, one could say that we are still in 2^{nd} wave orchestration technologies.

The works in Table 1 show how technologies are being integrated in pairs, with no efforts trying to cope with more than two at a time. This is a key characteristic of 2^{nd} wave orchestration technologies. Combining more than two orchestration comes with an exponential increase in complexity, which calls for new approaches towards comprehensive orchestration.

The next section presents some approaches to help us make the transition to 4^{th} wave orchestration faster and smoother.

4. New and Revisited Orchestration Approaches

4.1. Learning to Orchestrate

Machine learning (ML) techniques are starting to be applied to different aspects of the orchestration process in the cloud, such as data centre scheduling [148, 151, 152], IaaS instance selection [124], optimising resource scalability [153, 145, 147], network flow classification [154], network performance prediction [155], and software defect classification [156]. When fed with the enormous amount of logs kept by data centre and network operators, ML models are able to select the best configuration and location of resources.

These could be applied to making better and faster resource selection and configuration in edge/fog/IoT environments, but the amount and quality of data required and the need to pull these data out of many different organisations make it less workable, at least for now. Additionally, technologies for combining ML models trained for different domains (NFV in a telco with SDN in a data centre and SDN in a backbone network, for instance) into a single workable solution need to be explored.

The combination of models does not need to be in a hierarchical fashion [20, 89], however. For instance, in the case of a local ML model trained to optimise optical interconnects in the transport network and getting requests from a peer data centre model to open up connections with minimum latency for a set of VMs hosting a bulk data transfer or a WAN acceleration VNF. A transport network-associated neural network and the data centre-associated

Table 4: Pending challenges mapped to the requirements identified above (in brackets). * means it applies to all requirements above.

Pending Element	**Potential Solution**
Abstract failure **[churn, dynamism]**	QoS enabled deployments, describe tolerable availability
Data availability **[churn]**	Data-aware deployments [142]
Automated execution units description **[larger-scale, finer-grain]**	Automated software splitting [126]
Hyper- heterogeneity **[*]**	Self-describing components
Plain-English searches **[*]**	Information extraction and NLP
Device/service/function registries **[*]**	–
Data directory **[*]**	–
Matching resource requests and results **[dynamism,heterogeneity]**	Ontology-based searches [144]
Potentially O(N^2) negotiation	–
[churn, scale, multi-domain] [143]	Decentralised (P2P) registries
Registry scalability	creating libraries and packages of functions
discovery/sharing functionally similar functions	
Slow selection **[dynamism, speed, larger-scale]**	Improve selection based on previous runs [145]
Universal naming beyond LISP/ROC/MPTCP **[churn, multi-domain]**	Logical resource names [146]
Redundancy and "high availability" **[churn, dynamism]**	–
Edge-fog-cloud coordination **[*]**	
Orchestration always catching up **[churn, dynamism]**	Emergent behaviours, Asymptotic configurations [113, 58, 97, 132]
Automated adaptation **[heterogeneity]**	Unsupervised learning of configuration options
Isolation in FaaS/Serverless **[security]**	—
Cloud/Edge/Fog Security Coordination through	—
NFV/SDN (Trust, AAA, ChannelProtection, key-management) **[security]**	
Slow resource provisioning	
[large-scale, finer-grain]	
Deployment obsolescence	Predictive resource estimation [145, 147]
[dynamism, churn, larger-scale, finer-grain]	Predictive scheduling [148]
Stateful workflows	Delegation, asymptotic deployment [149, 132]
[dynamism, locality]	Data access prediction
Across provider federation	Workflow delegation/handover
[multi-domain, multi-org]	–
Accessing byte addressable memory beyond data centres	
[*]	
Data lifecycle management **[*]**	
Global workflow **[multi-domain, -org, larger-scale]**	Data-aware orchestrator
Balance synchronisation and network load	Delegation
[dynamism, speed, churn]	Statistical-monitoring -
Secure orchestration **[*]**	operate on aggregated monitoring information
Automated control loop-based monitoring/re-config [15]	Constant influx of security/privacy watchdog
Limited programming models	Automated loop constraint generation
[dynamism, data lifecycle, churn] [150, 99, 26]	HEBs [13]
Ability to debug and explain **[*]**	Abstract description and heavy delegation
Autonomic security reconfiguration and orchestration	–
in SDN-NFV-enabled Fog and IoT **[security]**	IA-driven contextual monitoring/reaction
Monitoring, accounting in FaaS/Serverless [9]	–
[security]	

neural network can automatically negotiate the best setup for that VNF based on prior instances of setting up that (or a similar) connection. These trained models can handle multiple objectives, like optimisation [58] and resiliency. Feature-Weighted Linear Stacking or buckets of models are commonly used techniques to combine models. We envision these neural net models can learn to talk to each other without specifying the details of the communication protocol (e.g. unsupervised learning of configuration protocols in Table 4).

Service discovery is more difficult to solve using ML techniques alone: services would need to be registered in some form of discovery service and tagged so that they can be found. This labelling needs to be done by highly qualified individuals, which makes the process tedious and not scalable.

Device churn, on the other hand, would require highly generalisable models, trained under an incredible variety of circumstances in order to minimise runtime overfitting-derived errors resulting from training with a very specific snapshot of the system.

While standards are needed, forcing humans to use these often results in no-standard usage or the development of yet another new "standard". Higher level languages are needed to solve this standard proliferation problem.

Declarative model-based languages focus on configuration of resources and services [113, 114, 92, 26], but they are less useful when it comes to service discovery and composition. Developers rely on the usage of web search engines to find compatible services. Information extraction, natural language processing, data and mining and similar technologies will help in this regard, as indicated in Table 4.

4.2. P2P/Agent-based Orchestration

P2P systems have traditionally excelled at delivering robust applications on vast numbers of edge devices with high volatility across multiple domains, but also within a single data centre for specific services (e.g. HPE Smart SAN[5]). P2P orchestration means independent agents that are capable of making autonomous decisions about a set of resources they control. These decisions are not necessarily prescribed by a set of immutable rules, but the agents adapt their strategies based on the state of the resources and the value of the applications they try to run on them.

Descriptions tend to be based on pre-established ontologies [144]. Thus, it is very complicated to make agent-based systems negotiate about a new type of request or resource they have never seen before (not in the ontology), requiring constant updates and maintenance. Hence, coping with hyper-heterogeneity still seems like a hard mission (Table 4).

In the case of a fog coalition where individual devices all negotiate how to orchestrate a running application, agents negotiating in pairs may take long (potential poor scalability, $O(N^2)$ for a full mesh – see Table 4) so some structure needs to be imposed to prevent long negotiation rounds. Moreover, high device churn would require negotiations to re-start. Also, discovering peers for negotiation can be complex under different domains [143].

A key goal for any orchestrator is to capitalise on low-level interfaces and synthesise new service-oriented abstractions that minimise human interaction and provision service in the order of minutes or seconds [20]. Taming service description and composition so that developers do not trade standardisation for convenience will drive a few research works in forthcoming years.

P2P orchestrators have also been tried as an alternative to centralised brokered orchestrators for quite a long time [157], but they have found very limited success in practice. For instance, Netflix have decided not to use P2P task choreography in their Conductor microservices orchestration engine. P2P systems tend to create implicit contracts that result in poorly documented tight coupling around input/output, SLAs, etc, making it harder to adapt to changing needs. Controlling the deployment/management of a myriad of individual controlling agents and creating a hierarchy for debugging/delegation/escalation is a complicated task (Table 4).

There is a wealth of knowledge about P2P/agent based systems and security, but some aspects introduced by the amalgamation of NFV/SDN/disaggregated data centres/FaaS are not well explored. Storing data from nearby devices may expose peers to legal liabilities that may hinder further developments.

4.3. Eventually Consistent/Probabilistic Orchestration

To ensure near-real-time intervention during IoT application development, one approach is to use correction mechanisms that could be iteratively applied even when suboptimal solutions are deployed initially. In this setting, the good

[5] `http://h20195.www2.hpe.com/V2/getpdf.aspx/a00001440enw.pdf`

old asymptotic and declarative management techniques [149, 132] may likely be applicable to manage these highly complex scenarios. In the same vein, differential consistency techniques, where devices get serializable consistency only in their neighbourhood (vs eventual consistency for further devices) have been suggested for distributed data stores [28]. Similarly, [15] suggest the use of several concurrent control loops that are automatically generated from a simple description language, as a mechanism to achieve eventual consistency between a desired state of the resources and their actual state (Table 4). Manual constraint generation no longer seems feasible in the light of current multi-domain, hyper-heterogeneous, IoT scale trends.

Massive scales, time uncertainty, resource dynamism, and delayed monitoring can all be coped with by applying asymptotic management techniques, as long as the application tolerates delays and performance does not degrade (or cost does not spike) quickly. Most current orchestration frameworks do not easily tolerate partial failure or undetermined delays to make resources gradually available and, thus, progressively take the system closer to the desired state (Table 4).

[158] provide a theoretical framework for the allocation of batch and service jobs in a set of constrained resources where some resources can be attacked or fail. Achieving a target reliability level can simply be a matter of placing extra replicas in different failure domains [159]. Defining failure domains at the edge, especially with end-user devices, can be difficult and requires proper observability and late characterisation of the failure modes of the devices. Also, resource definition languages need to be made in terms of tolerable availability and needed capacity, so that the orchestrator can factor these in. Orchestrators would also benefit from some historical knowledge to apply correction factors depending on previously seen churn and failure rates.

4.4. Hierarchical Delegation

The presence of a common data model (semantically rich enough for expressing the required goals) and a common mechanism for labelling the entities in the model (so that information can be fed backwards once a delegated operation has been materialised) enables this information exchange. Delegation approaches rely on declarative languages used by developers to express their resource needs [114].

Once the user specifies a set of elements to be deployed and how they are connected, the infrastructure labels each element with a unique name (Table 4). There is a degree of information that remains unknown for the user (e.g. underlying infrastructure details or topology of the virtual infrastructure, i.e. administrative domains and contract terms with each of these ones).

Some works use delegated workloads that are then analysed and scored before their allocation [160]. In spite of Google's workload and server heterogeneity, these are not comparable to the hyper-heterogeneity scenarios we described above. Also, dealing with orchestration of containers in a single domain helps make pragmatic decisions that work at scale in a well-confined administrative/security domain.

In a NFV/SDN/Serverless/Edge/Fog/IoT scenario descriptions are refined, transformed, and split (e.g. across several domains). It is possible that the configuration details for the edge nodes cannot be completed without information from the neighbouring domain. For example, they may need to exchange ports, IP address or tags depending on the nature of the connection. They may also need to agree which edge nodes to use. It is likely that this is information that they will not be willing to share with anyone other than their neighbouring domain. This implies the information will be obtained by interaction between providers and needs to be referenced differently as the model gets refined [146, 19].

Model refinement and splitting has a direct implication over the way things are referenced in the data model. For instance, when a user specifies she desires a VM in the UK and another one in the US, she is (likely) indirectly generating a split of her request across multiple administrative domains. When her request is split and refined, the VM in the UK is referring to a VM in the US whose final name (e.g. its host name) would only be known after deployment.

Hierarchical delegation models work well across administrative domains and their divide and conquer approach tends to enable larger scalability. Service discovery happens within a single administrative domain, where classic registry and search approaches have proven to work. However, implementing these systems in the light of hyper-heterogeneity is very difficult and expensive. They also fall short when it comes to coping with fluid dynamic organisations and high resource churn (Table 4).

4.5. No Orchestration

The best of orchestration might be having to do no orchestration at all. Large scale production systems call for very simple orchestration techniques where developers and operators can rapidly debug things gone badly (e.g. most

cloud schedulers use simple round robin for resource allocation). Even if the right abstractions are provided, building resource requests in high-level declarative languages can prove to be tedious and error prone for most developers. Several approaches try to bring the orchestration problem closer to developers and expose interfaces that let programmers specify behaviour, while concurrency and access control are individually dealt with by different devices (see Table 4).

[161] propose a framework where each device publishes a global log (time series of data and events) that is readily available for actuators to use. The framework is, however, too high level and does not really define how it would cope with trillions of time series.

Swarmlets are presented as an elegant way to use the actor model to wrap access to sensor devices [150]. Thus, developers would simply instantiate accessors on devices, reducing the orchestration needs. Either resources themselves or developers would have to control access, especially when modifying configurations. Indeed, it seems, Swarmlets add one level of indirection but similar questions as to how to publish, register and search for accessors; accessor security or lifecycle remain.

Fn Flow and PyWren [99] agree that the best approach to orchestrating FaaS is using familiar programming models (Java 8 lambdas and BSP or M/R, respectively). The description of the code functions would be done in the development environment and selection will come as using any library with dot autocomplete, leaving configuration, monitoring, and deployment to the underlying middleware. There are questions as how these libraries of functions could get organised and be made accessible for thousands of remote development environments without damaging developers' experience (see Table 4). There are also question marks on how this approach could cope with hyper-heterogeneity and edge deployments.

Beyond devices (or infrastructure) themselves, there is the problem of discovering many small fine-grained functions that could be (re)-used by multiple applications. There are no well-defined patterns for discovery across FaaS functions. While some of this is by no means FaaS specific the problem is exacerbated by the granular nature of FaaS functions and the lack of application / versioning definition. In this sense, building on classic software engineering practices (creating libraries and packages of functions) may be the way ahead (see Table 4).

Fog Dataflow programming frameworks have also recently appeared that support developers in dealing with scalability, heterogeneity, and mobility [162]. They do not support the levels of device autonomy, churn and hyper-heterogeneity (while keeping low levels of human intervention) we have described above.

Moreover, all of these approaches are very interesting for small homogeneous deployments confined to a single domain. The levels of complexity we will see in hyper-heterogeneous large scale fluid distributed organisations seem too complex for any single unassisted developer to cope with.

4.6. Security Orchestration

New context-aware holistic security orchestrators are needed to allow interfacing with NFV managers, SDN controllers and Edge-Fog controllers, thereby providing security chaining, as well as dynamic reconfiguration and adaptation of the virtual security appliances at the edge, in case of deviation from the expected behaviour.

In this sense, new security orchestration approaches are appearing recently. Open Security Controller (OSC)[163] is an open source project that tries to provide consistent security across a multi cloud environments. It aims to automate the deployment of virtualised network security functions to protect east-west traffic inside the data centre. It orchestrates the deployment of virtual network security policies, applying the correct policy to the appropriate workload.

Security Orchestrator [164] proposes a design of a Security Orchestrator in the context of the ETSI NFV Reference Architecture, defining the interfaces required to interact with the existing MANO entities. The Security orchestrator is placed outside the architecture to achieve a holistic end-to-end security view in case of a hybrid network.

In order to mitigate cyber-threats, latest research efforts focus on providing dynamic, intelligent and context-aware security orchestration in Fog/Edge and IoT by relying on NFV/SDN-enabled networks. This approach allows chaining and enforcing policy-based security mechanisms while providing run-time reconfiguration and adaptation of security enablers, and therefore, endowing the ecosystem with intelligent and dynamic behavior. In this sense, the H2020 EU project Anastacia [134] is also devising a security orchestrator to take autonomous decisions in MEC, Cloud and IoT scenarios, through the use of networking technologies such as SDN-NFV and intelligent and dynamic security policy enforcement and monitoring methodologies. In the Anastacia project, different virtual security appliances such as vFirewall, vIDS, vAAA, vSwitch/Router, vHoneynet, vVPN are orchestrated dynamically at the edge of the network.

In order to achieve a context-aware autonomic security orchestration in SDN/NFV-enabled Fog and IoT, we envisage the proliferation of cyber-situational awareness frameworks in which the security orchestration can dynamically be adapted according to the context obtained from agents and sentinels, mitigating and countering cyber security threats at the edge, by deploying and orchestrating Virtual Security Functions and services even in constrained Fog and IoT devices. Such awareness framework could be endowed with monitoring and reaction tools as well as innovative algorithms and techniques based on machine learning, for threat analysis, data fusion and correlation from different sources, and big data analysis. It would allow to increase the overall security, including self-repair, self-healing and self-protection capabilities, not only at the core, but also at the edge of the network.

4.7. Hierarchical Emergent Behaviours

A recent paper proposed Hierarchical Emergent Behaviours (HEB), an architecture that builds on established concepts of emergent behaviours and hierarchical decomposition and organisation. HEB's local rules induce emergent behaviours, i.e., useful behaviours not explicitly programmed [13] (see Table 4). This certainly is a promising approach, however it hinges heavily on the availability of an accurate and detailed model of all the resources available to an orchestrator, including all elements of the underlying infrastructure of available functions, resources, and deployment locations. Mechanisms to acquire such a model are not yet available in the literature, and is something that we hope to be closer to by addressing the challenges discussed thus far.

Emergent behaviours eliminate the need of a central orchestrator that would have to deal with a very large number of "things". They still require the use of high-level languages and associated tools (including ontologies) to describe the emergent behaviours [165, 166, 167, 168, 169].

Due to the large number of variables and situations, designing an explicit programmed system that takes into account all the scenarios in advance is a formidable task. With an HEB IoT approach and if the proper set of rules is defined, the "things" are able to dynamically adapt to the environment without the need to explicitly program them. However, they also extend the "attack surface" that can be exploited. An attacker that gains access could modify the rules, either directly or through modification of the hyper-parameters, for nefarious purposes.

5. Conclusions

Recent technological developments have been too quick for orchestration techniques to catch up. Most current orchestrators would fit in the 2^{nd} wave orchestration classification above, as they either do not cope with hardware programmability and disaggregation beyond a data centre.

Moreover, the amalgamation of many virtualised and disaggregated technologies is making end-to-end orchestration difficult to do at scale. This situation is getting worse with the advent of the IoT, where hyper-heterogeneity can make it nearly impossible for a single team to master all the knowledge needed across the stack. Scale is not the only problem, churn and dynamism makes it very hard to discover resources or plan how to synchronise as many of these devices connect to the network intermittently only and/or belong to different administrative domains.

Pushing current orchestrators to the next wave of maturity calls for further integration across (hardware) and "along" the technological spectrum these technologies cover. Recent works have made the challenges we identified less intimidating, offering promising results to tame data aware deployments, resource planning at scale (e.g. asymptotic plans), coping with churn (HEBs, logical naming, predictive scheduling) or resource searches (dynamic ontologies).

There is, however, lots of pending work to do if we aim for integrated solutions that can deliver a unified approach to orchestrate across technologies and administrative domains. Achieving 3^{rd} wave-level orchestrators requires better automation and complexity abstraction techniques and systems that can make automatic, but adaptive, decisions based on as few human inputs as possible. To a human developer or administrator, this mix of technologies needs to look as if it was a simple local deployment using a very declarative form, vs. very prescriptive specifications. Thus, artificial intelligence, NLP, HEBs, or "No orchestration" techniques can help smooth the interface between humans and this massive complexity, which are essential for 4^{th} wave-level orchestrators to become a reality.

Acknowledgements

The authors thank Gordon S Blair for his insightful comments on prior versions of this manuscript.

References

[1] L. M. Vaquero, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 27–32. doi:10.1145/2677046.2677052.
URL http://doi.acm.org/10.1145/2677046.2677052

[2] I. Azimi, A. Anzanpour, A. M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, N. Dutt, Hich: Hierarchical fog-assisted computing architecture for healthcare iot, ACM Trans. Embed. Comput. Syst. 16 (5s) (2017) 174:1–174:20. doi:10.1145/3126501.
URL http://doi.acm.org/10.1145/3126501

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proceedings of the IEEE 103 (1) (2015) 14–76. doi:10.1109/JPROC.2014.2371999.

[4] J. Gil Herrera, J. F. Botero, Resource allocation in nfv: A comprehensive survey, IEEE Trans. on Netw. and Serv. Manag. 13 (3) (2016) 518–532. doi:10.1109/TNSM.2016.2598420.
URL https://doi.org/10.1109/TNSM.2016.2598420

[5] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, S. Shenker, Network support for resource disaggregation in next-generation datacenters, in: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, ACM, New York, NY, USA, 2013, pp. 10:1–10:7. doi:10.1145/2535771.2535778.
URL http://doi.acm.org/10.1145/2535771.2535778

[6] A. Klimovic, C. Kozyrakis, E. Thereska, B. John, S. Kumar, Flash storage disaggregation, in: Proceedings of the Eleventh European Conference on Computer Systems, EuroSys '16, ACM, New York, NY, USA, 2016, pp. 29:1–29:15. doi:10.1145/2901318.2901337.
URL http://doi.acm.org/10.1145/2901318.2901337

[7] P. S. Rao, G. Porter, Is memory disaggregation feasible?: A case study with spark sql, in: Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems, ANCS '16, ACM, New York, NY, USA, 2016, pp. 75–80. doi:10.1145/2881025.2881030.
URL http://doi.acm.org/10.1145/2881025.2881030

[8] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, Serverless computation with openlambda, in: Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing, HotCloud'16, USENIX Association, Berkeley, CA, USA, 2016, pp. 33–39.
URL http://dl.acm.org/citation.cfm?id=3027041.3027047

[9] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, Serverless computing: Current trends and open problems, CoRR abs/1706.03178. arXiv:1706.03178.
URL http://arxiv.org/abs/1706.03178

[10] K. Liang, L. Zhao, X. Chu, H. H. Chen, An integrated architecture for software defined and virtualized radio access networks with fog computing, IEEE Network 31 (1) (2017) 80–87. doi:10.1109/MNET.2017.1600027NM.

[11] V. K. Sehgal, A. Patrick, A. Soni, L. Rajput, Smart Human Security Framework Using Internet of Things, Cloud and Fog Computing, Springer International Publishing, Cham, 2015, pp. 251–263.

[12] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, O. Fratu, Big data, internet of things and cloud convergence — an architecture for secure e-health applications, J. Med. Syst. 39 (11) (2015) 1–8. doi:10.1007/s10916-015-0327-y.
URL http://dx.doi.org/10.1007/s10916-015-0327-y

[13] D. Roca, D. Nemirovsky, M. Nemirovsky, R. Milito, M. Valero, Emergent behaviors in the internet of things: The ultimate ultra-large-scale system, IEEE Micro 36 (6) (2016) 36–44. doi:10.1109/MM.2016.102.

[14] A. Jonathan, A. Chandra, J. Weissman, Locality-aware load sharing in mobile cloud computing., in: In the IEEE/ACM International Conference on Utility and Cloud Computing (UCC), 2017.

[15] R. Abid, G. Salaün, N. D. Palma, Asynchronous synthesis techniques for coordinating autonomic managers in the cloud, Science of Computer Programming 146 (Supplement C) (2017) 87 – 103, special issue with extended selected papers from FACS 2015. doi:https://doi.org/10.1016/j.scico.2017.05.005.
URL http://www.sciencedirect.com/science/article/pii/S0167642317301089

[16] S. Cirani, G. Ferrari, N. Iotti, M. Picone, The iot hub: a fog node for seamless management of heterogeneous connected smart objects, in: International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops), 2015, pp. 1–6. doi:10.1109/SECONW.2015.7328145.

[17] V. Lopez, O. G. de Dios, L. Miguel, J. Foster, H. Silva, L. Blair, J. Marsella, T. Szyrkowiec, A. Autenrieth, C. Liou, A. Sadasivarao, S. Syed, J. Sunjun, B. Rao, F. Zhang, Demonstration of sdn orchestration in optical multi-vendor scenarios, in: Optical Fiber Communication Conference, Optical Society of America, 2015, p. Th2A.41. doi:10.1364/OFC.2015.Th2A.41.
URL http://www.osapublishing.org/abstract.cfm?URI=OFC-2015-Th2A.41

[18] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, Osmotic computing: A new paradigm for edge/cloud integration, IEEE Cloud Computing 3 (6) (2016) 76–83. doi:10.1109/MCC.2016.124.

[19] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, R. Muñoz, End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node, in: Optical Fiber Communications Conference and Exhibition (OFC), 2016, pp. 1–3.

[20] C. Rotsos, A. Farshad, N. Hart, A. Aguado, S. Bidkar, K. Sideris, D. King, L. Fawcett, J. Bird, A. Mauthe, N. Race, D. Hutchison, Baguette: Towards end-to-end service orchestration in heterogeneous networks, in: 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), 2016, pp. 196–203. doi:10.1109/IUCC-CSS.2016.035.

[21] A. Rostami, P. Öhlén, M. A. S. Santos, A. Vidal, Multi-domain orchestration across RAN and transport for 5G, in: Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM '16, ACM, New York, NY, USA, 2016, pp. 613–614. doi:10.1145/2934872.2959073.
URL http://doi.acm.org/10.1145/2934872.2959073

[22] P. Öhlén, B. Skubic, A. Rostami, M. Fiorani, P. Monti, Z. Ghebretensaé, J. Mårtensson, K. Wang, L. Wosinska, Data plane and control architectures for 5G transport networks, Journal of Lightwave Technology 34 (6) (2016) 1501–1508. doi:10.1109/JLT.2016.2524209.
[23] E. Yigitoglu, M. Mohamed, L. Liu, H. Ludwig, Foggy: A framework for continuous automated iot application deployment in fog computing, in: IEEE International Conference on AI Mobile Services (AIMS), 2017, pp. 38–45. doi:10.1109/AIMS.2017.14.
[24] M. Habib ur Rehman, P. P. Jayaraman, S. u. R. Malik, A. u. R. Khan, M. Medhat Gaber, Rededge: A novel architecture for big data processing in mobile edge computing environments, Journal of Sensor and Actuator Networks 6 (3).
URL http://www.mdpi.com/2224-2708/6/3/17
[25] N. B. Truong, G. M. Lee, Y. Ghamri-Doudane, Software defined networking-based vehicular adhoc network with fog computing, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1202–1207. doi:10.1109/INM.2015.7140467.
[26] C. Consel, M. Kabáč, Internet of things: From small- to large-scale orchestration, in: IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 1748–1755. doi:10.1109/ICDCS.2017.314.
[27] W. John, F. Moradi, B. Pechenot, P. Sköldström, Meeting the observability challenges for VNFs in 5G systems, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 1127–1130. doi:10.23919/INM.2017.7987445.
[28] R. Mayer, H. Gupta, E. Saurez, U. Ramachandran, Fogstore: Toward a distributed data store for fog computing, CoRR abs/1709.07558. arXiv:1709.07558.
URL http://arxiv.org/abs/1709.07558
[29] B. Martini, F. Paganelli, A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks, Future Internet 8 (2) (2016) 24. doi:10.3390/fi8020024.
URL https://doi.org/10.3390/fi8020024
[30] Y. Elkhatib, B. F. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, E. Rivière, On using micro-clouds to deliver the fog, Internet Computing 21 (2) (2017) 8–15. doi:10.1109/MIC.2017.35.
[31] A. G. Dalla-Costa, L. Bondan, J. A. Wickboldt, C. B. Both, L. Z. Granville, Maestro: An nfv orchestrator for wireless environments aware of vnf internal compositions, in: 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), 2017, pp. 484–491. doi:10.1109/AINA.2017.126.
[32] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, M. Rovatsos, Fog orchestration for internet of things services, IEEE Internet Computing 21 (2) (2017) 16–24. doi:10.1109/MIC.2017.36.
[33] D. Roca, J. V. Quiroga, M. Valero, M. Nemirovsky, Fog function virtualization: A flexible solution for iot applications, in: 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), 2017, pp. 74–80. doi:10.1109/FMEC.2017.7946411.
[34] A. Glikson, S. Nastic, S. Dustdar, Deviceless edge computing: Extending serverless computing to the edge of the network, in: Proceedings of the 10th ACM International Systems and Storage Conference, SYSTOR '17, ACM, New York, NY, USA, 2017, pp. 28:1–28:1. doi:10.1145/3078468.3078497.
URL http://doi.acm.org/10.1145/3078468.3078497
[35] T. B. Sousa, F. F. Correia, H. S. Ferreira, Patterns for software orchestration on the cloud, in: Proceedings of the 22nd Conference on Pattern Languages of Programs, PLoP '15, The Hillside Group, USA, 2015, pp. 17:1–17:12.
URL http://dl.acm.org/citation.cfm?id=3124497.3124517
[36] Y. Jiang, Z. Huang, D. H. K. Tsang, Challenges and solutions in fog computing orchestration, IEEE Network PP (99) (2017) 1–8. doi:10.1109/MNET.2017.1700271.
[37] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, X. Shen, Internet of vehicles in big data era, IEEE/CAA Journal of Automatica Sinica 5 (1) (2018) 19–35. doi:10.1109/JAS.2017.7510736.
[38] OpenFog reference architecture for fog computing, last accessed: Dec 2017 (2017).
URL https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf
[39] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, T. F. Wenisch, Disaggregated memory for expansion and sharing in blade servers, SIGARCH Comput. Archit. News 37 (3) (2009) 267–278. doi:10.1145/1555815.1555789.
URL http://doi.acm.org/10.1145/1555815.1555789
[40] L. A. Barroso, U. Hoelzle, The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines, 1st Edition, Morgan and Claypool Publishers, 2009.
[41] G. Coglitore, A. Michael, J. Williams, M. Corddry, Disaggregation of server components in a data center, uS Patent App. 13/709,004 (Jun. 12 2014).
URL https://www.google.com/patents/US20140164669
[42] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, S. Shenker, Network requirements for resource disaggregation, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, USENIX Association, Berkeley, CA, USA, 2016, pp. 249–264.
URL http://dl.acm.org/citation.cfm?id=3026877.3026897
[43] C. Kilcioglu, J. M. Rao, A. Kannan, R. P. McAfee, Usage patterns and the economics of the public cloud, in: Proceedings of the 26th International Conference on World Wide Web, WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2017, pp. 83–91. doi:10.1145/3038912.3052707.
URL https://doi.org/10.1145/3038912.3052707
[44] S. Legtchenko, H. Williams, K. Razavi, A. Donnelly, R. Black, A. Douglas, N. Cheriere, D. Fryer, K. Mast, A. D. Brown, A. Klimovic, A. Slowey, A. Rowstron, Understanding rack-scale disaggregated storage, in: 9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17), USENIX Association, Santa Clara, CA, 2017.
URL https://www.usenix.org/conference/hotstorage17/program/presentation/legtchenko
[45] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, K. G. Shin, Efficient memory disaggregation with infiniswap, in: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), USENIX Association, Boston, MA, 2017, pp. 649–667.
URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/gu

[46] B. Caldwell, Y. Im, S. Ha, R. Han, E. Keller, Fluidmem: Memory as a service for the datacenter, CoRR abs/1707.07780. arXiv:1707.07780.
URL http://arxiv.org/abs/1707.07780

[47] M. K. Aguilera, N. Amit, I. Calciu, X. Deguillard, J. Gandhi, P. Subrahmanyam, L. Suresh, K. Tati, R. Venkatasubramanian, M. Wei, Remote memory in the age of fast networks, in: Proceedings of the 2017 Symposium on Cloud Computing, SoCC '17, ACM, New York, NY, USA, 2017, pp. 121–127. doi:10.1145/3127479.3131612.
URL http://doi.acm.org/10.1145/3127479.3131612

[48] K. Lim, Y. Turner, J. R. Santos, A. AuYoung, J. Chang, P. Ranganathan, T. F. Wenisch, System-level implications of disaggregated memory, in: Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture, HPCA '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 1–12. doi:10.1109/HPCA.2012.6168955.
URL http://dx.doi.org/10.1109/HPCA.2012.6168955

[49] D. Syrivelis, A. Reale, K. Katrinis, I. Syrigos, M. Bielski, D. Theodoropoulos, D. Pnevmatikatos, G. Zervas, A software-defined architecture and prototype for disaggregated memory rack scale systems, in: Proceedings of the Seventeenth IEEE International Conference on Embedded Computer Systems: Architectures, Modelling, and Simulation (SAMOS XVII), SAMOS '17, IEEE, 2017.

[50] CCIX: cache coherent interconnect for accelerators, last accessed: Dec 2017 (Dec. 12 2017).
URL http://www.ccixconsortium.com

[51] Gen-Z draft core specification, last accessed: Dec 2017 (Dec. 12 2016).
URL http://genzconsortium.org/

[52] Opencapi specification, last accessed: Dec 2017 (Dec. 12 2016).
URL http://opencapi.org

[53] Intel omni-path, last accessed: Dec 2017 (Dec. 12 2017).
URL http://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-architecture-fabric-overview.html

[54] J. Gray, Tape is dead disk is tape flash is disk ram locality is king, last accessed: Dec 2017 (2006).
URL https://jimgray.azurewebsites.net/talks/Flash_is_Good.ppt

[55] M. A. Sevilla, N. Watkins, I. Jimenez, P. Alvaro, S. Finkelstein, J. LeFevre, C. Maltzahn, Malacology: A programmable storage system, in: Proceedings of the Twelfth European Conference on Computer Systems, EuroSys '17, ACM, New York, NY, USA, 2017, pp. 175–190. doi:10.1145/3064176.3064208.
URL http://doi.acm.org/10.1145/3064176.3064208

[56] I. Stefanovici, B. Schroeder, G. O'Shea, E. Thereska, Treating the storage stack like a network, Trans. Storage 13 (1) (2017) 2:1–2:27. doi:10.1145/3032968.
URL http://doi.acm.org/10.1145/3032968

[57] P. Faraboschi, K. Keeton, T. Marsland, D. Milojicic, Beyond processor-centric operating systems, in: Proceedings of the 15th USENIX Conference on Hot Topics in Operating Systems, HOTOS'15, USENIX Association, Berkeley, CA, USA, 2015, pp. 17–17.
URL http://dl.acm.org/citation.cfm?id=2831090.2831107

[58] N. Huin, B. Jaumard, F. Giroire, Optimization of network service chain provisioning, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–7. doi:10.1109/ICC.2017.7997198.

[59] R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, G. Mastorakis, Drop computing: Ad-hoc dynamic collaborative computing, Future Generation Computer Systems (2017) –doi:https://doi.org/10.1016/j.future.2017.11.044.
URL https://www.sciencedirect.com/science/article/pii/S0167739X17305678

[60] S. S. Lor, L. M. Vaquero, P. Murray, In-NetDC: The cloud in core networks, IEEE Communications Letters 16 (10) (2012) 1703–1706. doi:10.1109/LCOMM.2012.090312.121543.

[61] Aws greengrass, last accessed: Dec 2017 (2016).
URL https://aws.amazon.com/greengrass/

[62] Microsoft azure IoT edge, last accessed: Dec 2017 (2016).
URL https://azure.microsoft.com/en-in/campaigns/iot-edge/

[63] Apache kura, last accessed: Dec 2017 (2017).
URL https://www.eclipse.org/kura/

[64] VMWare's apache liota, last accessed: Dec 2017 (2017).
URL https://github.com/vmware/liota

[65] N. Wang, B. Varghese, M. Matthaiou, D. S. Nikolopoulos, Enorm: A framework for edge node resource management, IEEE Transactions on Services Computing PP (99) (2017) 1–1. doi:10.1109/TSC.2017.2753775.

[66] M. Liyanage, C. Chang, S. N. Srirama, mepaas: Mobile-embedded platform as a service for distributing fog computing to edge nodes, in: Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2016 17th International Conference on, IEEE, 2016, pp. 73–80.

[67] N. K. Giang, M. Blackstock, R. Lea, V. C. Leung, Developing IoT applications in the fog: A distributed dataflow approach, in: 5th International Conference on the Internet of Things (IOT), IEEE, 2015, pp. 155–162.

[68] P. Ravindra, A. Khochare, S. Reddy, S. Sharma, P. Varshney, Y. Simmhan, ECHO: An adaptive orchestration platform for hybrid dataflows across cloud and edge, International Conference on Service-Oriented Computing (ICSOC).
URL http://arxiv.org/abs/1707.00889

[69] O. Skarlat, S. Schulte, M. Borkowski, P. Leitner, Resource provisioning for iot services in the fog, in: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), 2016, pp. 32–39. doi:10.1109/SOCA.2016.10.

[70] Y. Elkhatib, Mapping Cross-Cloud Systems: Challenges and Opportunities, in: Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing, USENIX Association, 2016, pp. 77–83.

[71] M. S. de Brito, S. Hoque, T. Magedanz, R. Steinke, A. Willner, D. Nehls, O. Keils, F. Schreiner, A service orchestration architecture

for fog-enabled infrastructures, in: Second International Conference on Fog and Mobile Edge Computing (FMEC), 2017, pp. 127–132. doi:10.1109/FMEC.2017.7946419.

[72] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, MCC '13, ACM, New York, NY, USA, 2013, pp. 15–20. doi:10.1145/2491266.2491270.
URL http://doi.acm.org/10.1145/2491266.2491270

[73] Y. Elkhatib, Building cloud applications for challenged networks, in: R. Horne (Ed.), Embracing Global Computing in Emerging Economies, Vol. 514 of Communications in Computer and Information Science, Springer International Publishing, 2015, pp. 1–10. doi:10.1007/978-3-319-25043-4_1.
URL http://dx.doi.org/10.1007/978-3-319-25043-4_1

[74] R. Guerzoni, I. Vaishnavi, D. Perez Caparros, A. Galis, F. Tusa, P. Monti, A. Sganbelluri, G. Biczók, B. Sonkoly, L. Toka, A. Ramos, J. Melián, O. Dugeon, F. Cugini, B. Martini, P. Iovanna, G. Giuliani, R. Figueiredo, L. M. Contreras-Murillo, C. J. Bernardos, C. Santana, R. Szabo, Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey, Transactions on Emerging Telecommunications Technologies 28 (4) (2017) e3103. doi:10.1002/ett.3103.
URL http://dx.doi.org/10.1002/ett.3103

[75] M. Satyanarayanan, A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets, GetMobile: Mobile Comp. and Comm. 18 (4) (2015) 19–23. doi:10.1145/2721914.2721921.
URL http://doi.acm.org/10.1145/2721914.2721921

[76] R. Roman, J. Zhou, J. Lopez, On the features and challenges of security and privacy in distributed internet of things, Computer Networks 57 (10) (2013) 2266 – 2279, towards a Science of Cyber Security Security and Identity Architecture for the Future Internet. doi:https://doi.org/10.1016/j.comnet.2012.12.018.
URL http://www.sciencedirect.com/science/article/pii/S1389128613000054

[77] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, in: Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, IEEE, 2014, pp. 1–8.

[78] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. K. R. Choo, M. Dlodlo, From cloud to fog computing: A review and a conceptual live VM migration framework, IEEE Access 5 (2017) 8284–8300. doi:10.1109/ACCESS.2017.2692960.

[79] R. Roman, J. Lopez, M. Mambo, Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges, Future Generation Computer Systems 78 (Part 2) (2018) 680 – 698. doi:https://doi.org/10.1016/j.future.2016.11.009.
URL http://www.sciencedirect.com/science/article/pii/S0167739X16305635

[80] M. Haus, M. Waqas, A. Y. Ding, Y. Li, S. Tarkoma, J. Ott, Security and privacy in device-to-device (d2d) communication: A review, IEEE Communications Surveys Tutorials 19 (2) (2017) 1054–1079. doi:10.1109/COMST.2017.2649687.

[81] J. B. Bernabé, J. L. H. Ramos, A. F. Gómez-Skarmeta, Holistic privacy-preserving identity management system for the internet of things, Mobile Information Systems 2017 (2017) 6384186:1–6384186:20.

[82] J. Bernal Bernabe, J. L. Hernandez Ramos, A. F. Skarmeta Gomez, Taciot: multidimensional trust-aware access control system for the internet of things, Soft Computing 20 (5) (2016) 1763–1779. doi:10.1007/s00500-015-1705-6.
URL https://doi.org/10.1007/s00500-015-1705-6

[83] R. Lu, X. Liang, X. Li, X. Lin, X. Shen, Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications, IEEE Transactions on Parallel and Distributed Systems 23 (9) (2012) 1621–1631.

[84] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, D. Lopez, Management and orchestration challenges in network functions virtualization, IEEE Communications Magazine 54 (1) (2016) 98–105. doi:10.1109/MCOM.2016.7378433.

[85] K. Katsalis, N. Nikaein, A. Edmonds, Multi-domain orchestration for nfv: Challenges and research directions, in: 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), 2016, pp. 189–195. doi:10.1109/IUCC-CSS.2016.034.

[86] R. Mijumbi, J. Serrat, J. L. Gorricho, J. Rubio-Loyola, S. Davy, Server placement and assignment in virtualized radio access networks, in: 2015 11th International Conference on Network and Service Management (CNSM), 2015, pp. 398–401. doi:10.1109/CNSM.2015.7367390.

[87] R. Munoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. Lopez, D. Lopez, Integrated sdn/nfv management and orchestration architecture for dynamic deployment of virtual sdn control instances for virtual tenant networks [invited], IEEE/OSA Journal of Optical Communications and Networking 7 (11) (2015) B62–B70. doi:10.1364/JOCN.7.000B62.

[88] S. J. Vaughan-Nichols, Virtualization sparks security concerns, Computer 41 (8) (2008) 13–15. doi:10.1109/MC.2008.276.

[89] C. Rotsos, D. King, A. Farshad, J. Bird, L. Fawcett, N. Georgalas, M. Gunkel, K. Shiomoto, A. Wang, A. Mauthe, N. Race, D. Hutchison, Network service orchestration standardization, Comput. Stand. Interfaces 54 (P4) (2017) 203–215. doi:10.1016/j.csi.2016.12.006.
URL https://doi.org/10.1016/j.csi.2016.12.006

[90] Network functions virtualisation (nfv);nfv security;problem statement, Technical report, Industry Specification Group (ISG) Network Functions Virtualisation (NFV). (2014).

[91] S. Scott-Hayward, G. O'Callaghan, S. Sezer, Sdn security: A survey, in: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–7. doi:10.1109/SDN4FNS.2013.6702553.

[92] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, G. Wang, Meridian: an sdn platform for cloud network services, IEEE Communications Magazine 51 (2) (2013) 120–127. doi:10.1109/MCOM.2013.6461196.

[93] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, D. Soldani, A novel approach to virtual networks embedding for SDN management and orchestration, in: 2014 IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–7. doi:10.1109/NOMS.2014.6838244.

[94] e. a. Marie-Paule Odini, Report on sdn usage in nfv architectural framework, Technical report, ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV) (2015).

[95] S. Shin, G. Gu, Attacking software-defined networks: A first feasibility study, in: Proceedings of the Second ACM SIGCOMM Workshop

on Hot Topics in Software Defined Networking, HotSDN '13, ACM, New York, NY, USA, 2013, pp. 165–166. doi:10.1145/2491185.2491220.
URL http://doi.acm.org/10.1145/2491185.2491220

[96] S. Pradhan, A. Dubey, S. Khare, S. Nannapaneni, A. Gokhale, S. Mahadevan, D. C. Schmidt, M. Lehofer, Chariot: A holistic, goal driven orchestration solution for resilient iot applications, ACM Transactions on Cyber-Physical Systems.

[97] A. Amjad, F. Rabby, S. Sadia, M. Patwary, E. Benkhelifa, Cognitive edge computing based resource allocation framework for internet of things, in: Second International Conference on Fog and Mobile Edge Computing (FMEC), 2017, pp. 194–200. doi:10.1109/FMEC.2017.7946430.

[98] C. Chang, S. N. Srirama, R. Buyya, Mobile cloud business process management system for the internet of things: a survey, ACM Computing Surveys (CSUR) 49 (4) (2017) 70.

[99] E. Jonas, S. Venkataraman, I. Stoica, B. Recht, Occupy the cloud: Distributed computing for the 99%, CoRR abs/1702.04024. arXiv:1702.04024.
URL http://arxiv.org/abs/1702.04024

[100] G. Podjarny, Serverless security implications—from infra to owasp, last accessed: Dec 2017 (2017).
URL https://snyk.io/blog/serverless-security-implications-from-infra-to-owasp/

[101] AWS step functions, last accessed: Dec 2017 (2017).
URL https://aws.amazon.com/step-functions/

[102] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, in: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6, OSDI'04, USENIX Association, Berkeley, CA, USA, 2004, pp. 10–10.
URL http://dl.acm.org/citation.cfm?id=1251254.1251264

[103] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 10–10.
URL http://dl.acm.org/citation.cfm?id=1863103.1863113

[104] Gs nfv-man 001 - v1.1.1 - network functions virtualisation (nfv); management and orchestration, last accessed: Dec 2017 (2014).
URL http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[105] ETSI mobile edge computing: Framework and reference architecture, last accessed: Dec 2017 (2014).
URL http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf

[106] Sdn architecture, last accessed: Dec 2017 (2016).
URL https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf

[107] Topology and orchestration specification for cloud applications (TOSCA), last accessed: Dec 2017 (2013).
URL http://docs.oasis-open.org/tosca/tosca-primer/v1.0/tosca-primer-v1.0.html

[108] L. Velasco, A. Castro, D. King, O. Gerstel, R. Casellas, V. Lopez, In-operation network planning, IEEE Communications Magazine 52 (1) (2014) 52–60. doi:10.1109/MCOM.2014.6710064.

[109] S. Nadgowda, S. Suneja, C. Isci, Paracloud: Bringing application insight into cloud operations, in: 9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17), USENIX Association, Santa Clara, CA, 2017.
URL https://www.usenix.org/conference/hotcloud17/program/presentation/nadgowda

[110] C. Peltz, Web services orchestration and choreography, Computer 36 (10) (2003) 46–52. doi:10.1109/MC.2003.1236471.
URL http://dx.doi.org/10.1109/MC.2003.1236471

[111] H. Muñoz Frutos, I. Kotsiopoulos, L. M. Vaquero Gonzalez, L. Rodero Merino, Enhancing Service Selection by Semantic QoS, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 565–577.

[112] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for vm-based cloudlets in mobile computing, IEEE Pervasive Computing 8 (4) (2009) 14–23. doi:10.1109/MPRV.2009.82.

[113] P. Goldsack, J. Guijarro, A. Lain, G. Mecheneau, P. Murray, P. Toft, Smartfrog: Configuration and automatic ignition of distributed applications, in: In: HP Openview University Association Conference (HP OVUA, 2003, pp. 1–9.

[114] I. Bokharouss, Gcl viewer: A study in improving the understanding of gcl programs, last accessed: Dec 2017 (2008).
URL http://repository.tue.nl/638953

[115] B. Liskov, Practical uses of synchronized clocks in distributed systems, Distrib. Comput. 6 (4) (1993) 211–219. doi:10.1007/BF02242709.
URL http://dx.doi.org/10.1007/BF02242709

[116] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al., Spanner: Google's globally distributed database, ACM Transactions on Computer Systems (TOCS) 31 (3) (2013) 8.

[117] V. Sharma, K. Srinivasan, D. N. K. Jayakody, O. F. Rana, R. Kumar, Managing service-heterogeneity using osmotic computing, CoRR abs/1704.04213. arXiv:1704.04213.
URL http://arxiv.org/abs/1704.04213

[118] H. Karl, S. Dräxler, M. Peuster, A. Galis, M. Bredel, A. Ramos, J. Martrat, M. S. Siddiqui, S. van Rossem, W. Tavernier, G. Xilouris, Devops for network function virtualisation: an architectural approach, Transactions on Emerging Telecommunications Technologies 27 (9) (2016) 1206–1215, ett.3084. doi:10.1002/ett.3084.
URL http://dx.doi.org/10.1002/ett.3084

[119] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, M. Nemirovsky, Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing, in: 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014, pp. 325–329. doi:10.1109/CAMAD.2014.7033259.

[120] N. Mohan, P. Zhou, K. Govindaraj, J. Kangasharju, Managing data in computational edge clouds, in: Proceedings of the Workshop on Mobile Edge Communications, MECOMM '17, ACM, New York, NY, USA, 2017, pp. 19–24. doi:10.1145/3098208.3098212.
URL http://doi.acm.org/10.1145/3098208.3098212

[121] R. Bonafiglia, G. Castellano, I. Cerrato, F. Risso, End-to-end service orchestration across sdn and cloud computing domains, in: 2017 IEEE

Conference on Network Softwarization (NetSoft), 2017, pp. 1–6. doi:10.1109/NETSOFT.2017.8004234.
[122] K. Nahrstedt, J. M. Smith, The qos broker [distributed multimedia computing], IEEE MultiMedia 2 (1) (1995) 53–67. doi:10.1109/93.368603.
[123] F. Jrad, J. Tao, A. Streit, A broker-based framework for multi-cloud workflows, in: Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds, MultiCloud '13, ACM, New York, NY, USA, 2013, pp. 61–68. doi:10.1145/2462326.2462339.
URL http://doi.acm.org/10.1145/2462326.2462339
[124] F. Samreen, Y. Elkhatib, M. Rowe, G. S. Blair, Daleel: Simplifying cloud instance selection using machine learning, in: Proceedings of the Network Operations and Management Symposium, IEEE, 2016, pp. 557–563. doi:10.1109/NOMS.2016.7502858.
[125] S. Latre, J. Famaey, F. D. Turck, P. Demeester, The fluid internet: service-centric management of a virtualized future internet, IEEE Communications Magazine 52 (1) (2014) 140–148. doi:10.1109/MCOM.2014.6710076.
[126] J. Spillner, S. Dorodko, Java code analysis and transformation into AWS lambda functions, CoRR abs/1702.05510. arXiv:1702.05510.
URL http://arxiv.org/abs/1702.05510
[127] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, in: 2014 Federated Conference on Computer Science and Information Systems, 2014, pp. 1–8. doi:10.15439/2014F503.
[128] System architecture specification for execution of sensitive nfv components, Technical report, ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV) (2017).
[129] Y. W. Law, M. Palaniswami, G. Kounga, A. Lo, WAKE: Key management scheme for wide-area measurement systems in smart grid, IEEE Communications Magazine 51 (1) (2013) 34–41. doi:10.1109/MCOM.2013.6400436.
[130] Y. Sun, S. Nanda, T. Jaeger, Security-as-a-service for microservices-based cloud applications, in: 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), 2015, pp. 50–57. doi:10.1109/CloudCom.2015.93.
[131] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, A. Rindos, A novel framework for software defined based secure storage systems, Simulation Modelling Practice and Theory 77 (Supplement C) (2017) 407 – 423. doi:https://doi.org/10.1016/j.simpat.2016.05.003.
URL http://www.sciencedirect.com/science/article/pii/S1569190X15303014
[132] P. Anderson, G. Beckett, K. Kavoussanakis, G. Mecheneau, P. Toft, Technologies for large-scale configuration management. the gridweaver project, Technical report, University of Glasgow, Glasgow, UK (2002).
URL http://www.gridweaver.org/WP1/report1.pdf
[133] D. Tuncer, M. Charalambides, S. Clayman, G. Pavlou, Adaptive resource management and control in software defined networks, IEEE Transactions on Network and Service Management 12 (1) (2015) 18–33. doi:10.1109/TNSM.2015.2402752.
[134] S. Ziegler, A. Skarmeta, J. Bernal, E. Kim, S. Bianchi, Anastacia: Advanced networked agents for security and trust assessment in cps iot architectures, in: 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1–6. doi:10.1109/GIOTS.2017.8016285.
[135] J. P. Santos, R. Alheiro, L. Andrade, V. Caraguay, Á. Leonardo, L. I. Barona López, M. A. Sotelo Monge, L. J. Garcia Villalba, W. Jiang, H. Schotten, et al., Selfnet framework self-healing capabilities for 5g mobile networks, Transactions on Emerging Telecommunications Technologies 27 (9) (2016) 1225–1232.
[136] T. Xu, D. Gao, P. Dong, H. Zhang, C. H. Foh, H. C. Chao, Defending against new-flow attack in sdn-based internet of things, IEEE Access 5 (2017) 3431–3443. doi:10.1109/ACCESS.2017.2666270.
[137] N. Bila, P. Dettori, A. Kanso, Y. Watanabe, A. Youssef, Leveraging the serverless architecture for securing linux containers, in: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017, pp. 401–404. doi:10.1109/ICDCSW.2017.66.
[138] S. J. Stolfo, M. B. Salem, A. D. Keromytis, Fog computing: Mitigating insider data theft attacks in the cloud, in: IEEE Symposium on Security and Privacy Workshops, 2012, pp. 125–128. doi:10.1109/SPW.2012.19.
[139] R. Lu, K. Heung, A. H. Lashkari, A. A. Ghorbani, A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot, IEEE Access 5 (2017) 3302–3312. doi:10.1109/ACCESS.2017.2677520.
[140] Y. Gao, Y. Peng, F. Xie, W. Zhao, D. Wang, X. Han, T. Lu, Z. Li, Analysis of security threats and vulnerability for cyber-physical systems, in: Proceedings of 2013 3rd International Conference on Computer Science and Network Technology, 2013, pp. 50–55. doi:10.1109/ICCSNT.2013.6967062.
[141] J. B. Bernabe, G. M. Perez, A. F. Skarmeta Gomez, Intercloud trust and security decision support system: an ontology-based approach, Journal of Grid Computing 13 (3) (2015) 425–456. doi:10.1007/s10723-015-9346-7.
URL https://doi.org/10.1007/s10723-015-9346-7
[142] A. L. Lemos, F. Daniel, B. Benatallah, Web service composition: A survey of techniques and tools, ACM Comput. Surv. 48 (3) (2015) 33:1–33:41. doi:10.1145/2831270.
URL http://doi.acm.org/10.1145/2831270
[143] M. Kelaskar, V. Matossian, P. Mehra, D. Paul, M. Parashar, A study of discovery mechanisms for peer-to-peer applications, in: Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, 2002, pp. 444–444. doi:10.1109/CCGRID.2002.1017187.
[144] K. M. Sim, Agent-based cloud computing, IEEE Transactions on Services Computing 5 (4) (2012) 564–577. doi:10.1109/TSC.2011.52.
[145] A. A. Bankole, S. A. Ajila, Predicting cloud resource provisioning using machine learning techniques, in: 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2013, pp. 1–4. doi:10.1109/CCECE.2013.6567848.
[146] B. M. et al., Cloud network architecture description in scalable and adaptable internet solutions, Technical report, EU, Brussls, EU (2001).
URL http://www.sail-project.eu/wp-content/uploads/2011/09/SAIL_DD1_final_public.pdf
[147] H. Arabnejad, C. Pahl, P. Jamshidi, G. Estrada, A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling, in: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '17, IEEE Press, Piscataway, NJ, USA, 2017, pp. 64–73. doi:10.1109/CCGRID.2017.15.
URL https://doi.org/10.1109/CCGRID.2017.15
[148] J. L. Berral, I. n. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, J. Torres, Towards energy-aware scheduling in data centers using machine

learning, in: International Conference on Energy-Efficient Computing and Networking, e-Energy '10, ACM, New York, NY, USA, 2010, pp. 215–224. doi:10.1145/1791314.1791349.
URL http://doi.acm.org/10.1145/1791314.1791349

[149] G. Pollock, D. Thompson, J. Sventek, P. Goldsack, The asymptotic configuration of application components in a distributed system, Technical report, University of Glasgow, Glasgow, UK (1998).
URL http://eprints.gla.ac.uk/79048/

[150] E. Latronico, E. A. Lee, M. Lohstroh, C. Shaver, A. Wasicek, M. Weber, A vision of swarmlets, IEEE Internet Computing 19 (2) (2015) 20–28. doi:10.1109/MIC.2015.17.

[151] J. Gao, R. Jamidar, Machine learning applications for data center optimization (2014).

[152] V. S. Marco, B. Taylor, B. Porter, Z. Wang, Improving spark application throughput via memory aware task co-location: A mixture of experts approach, in: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, Middleware '17, ACM, New York, NY, USA, 2017, pp. 95–108. doi:10.1145/3135974.3135984.
URL http://doi.acm.org/10.1145/3135974.3135984

[153] P. Bodík, R. Griffith, C. Sutton, A. Fox, M. Jordan, D. Patterson, Statistical machine learning makes automatic control practical for internet datacenters, in: Conference on Hot Topics in Cloud Computing, HotCloud'09, USENIX Association, Berkeley, CA, USA, 2009.
URL http://dl.acm.org/citation.cfm?id=1855533.1855545

[154] S. Zander, T. Nguyen, G. Armitage, Automated traffic classification and application identification using machine learning, in: IEEE Conference on Local Computer Networks 30th Anniversary, LCN '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 250–257. doi:10.1109/LCN.2005.35.
URL https://doi.org/10.1109/LCN.2005.35

[155] R. Stadler, R. Pasquini, V. Fodor, Learning from network device statistics, Journal of Network and Systems Management 25 (4) (2017) 672–698. doi:10.1007/s10922-017-9426-z.
URL https://doi.org/10.1007/s10922-017-9426-z

[156] J. Nam, S. Kim, Clami: Defect prediction on unlabeled datasets (t), in: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2015, pp. 452–463. doi:10.1109/ASE.2015.56.

[157] G. B. Chafle, S. Chandra, V. Mann, M. G. Nanda, Decentralized orchestration of composite web services, in: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters, WWW Alt. '04, ACM, New York, NY, USA, 2004, pp. 134–143. doi:10.1145/1013367.1013390.
URL http://doi.acm.org/10.1145/1013367.1013390

[158] M. Korupolu, R. Rajaraman, Robust and probabilistic failure-aware placement, in: 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '16, ACM, New York, NY, USA, 2016, pp. 213–224. doi:10.1145/2935764.2935802.
URL http://doi.acm.org/10.1145/2935764.2935802

[159] M. Sedaghat, E. Wadbro, J. Wilkes, S. D. Luna, O. Seleznjev, E. Elmroth, Diehard: Reliable scheduling to survive correlated failures in cloud data centers, in: 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 52–59. doi:10.1109/CCGrid.2016.11.

[160] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, J. Wilkes, Large-scale cluster management at Google with Borg, in: European Conference on Computer Systems (EuroSys), Bordeaux, France, 2015.

[161] B. Zhang, N. Mor, J. Kolb, D. S. Chan, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, J. Kubiatowicz, The cloud is not enough: Saving iot from the cloud, in: 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15), USENIX Association, Santa Clara, CA, 2015.
URL https://www.usenix.org/conference/hotcloud15/workshop-program/presentation/zhang

[162] N. K. Giang, M. Blackstock, R. Lea, V. C. M. Leung, Developing iot applications in the fog: A distributed dataflow approach, in: 2015 5th International Conference on the Internet of Things (IOT), 2015, pp. 155–162. doi:10.1109/IOT.2015.7356560.

[163] T. L. Foundation, Open security controller, last accessed: Dec 2017 (2017).
URL https://www.opensecuritycontroller.org

[164] B. Jaeger, Security orchestrator: Introducing a security orchestrator in the context of the etsi nfv reference architecture, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, 2015, pp. 1255–1260. doi:10.1109/Trustcom.2015.514.

[165] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, E. Jansen, The gator tech smart house: a programmable pervasive space, Computer 38 (3) (2005) 50–60. doi:10.1109/MC.2005.107.

[166] G. S. Blair, Y.-D. Bromberg, G. Coulson, Y. Elkhatib, L. Réveillère, H. B. Ribeiro, E. Rivière, F. Taïani, Holons: Towards a systematic approach to composing systems of systems, in: Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware, ARM 2015, ACM, New York, NY, USA, 2015, pp. 5:1–5:6. doi:10.1145/2834965.2834970.
URL http://doi.acm.org/10.1145/2834965.2834970

[167] G. S. Blair, D. Schmidt, C. Taconet, Middleware for internet distribution in the context of cloud computing and the internet of things, Annals of Telecommunications 71 (3) (2016) 87–92. doi:10.1007/s12243-016-0493-z.
URL https://doi.org/10.1007/s12243-016-0493-z

[168] A. Diaconescu, S. Frey, C. Müller-Schloer, J. Pitt, S. Tomforde, Goal-oriented holonics for complex system (self-)integration: Concepts and case studies, in: IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2016, pp. 100–109. doi:10.1109/SASO.2016.16.

[169] F. Oquendo, Architecturally describing the emergent behavior of software-intensive system-of-systems with sosadl, in: 12th System of Systems Engineering Conference (SoSE), 2017, pp. 1–6. doi:10.1109/SYSOSE.2017.7994941.