











Integrating SDN and NFV with QoS-Aware Service Composition

Valeria Cardellini¹ , Tihana Galinac Grbac² , Andreas Kassler³ ,
Pradeeban Kathiravelu⁴ , Francesco Lo Presti¹ , Antonio Marotta³ ,
Matteo Nardelli¹ , and Luís Veiga⁴ 

¹ University of Rome Tor Vergata, Rome, Italy
{cardellini,nardelli}@ing.uniroma2.it, lopresti@info.uniroma2.it

² Faculty of Engineering, University of Rijeka, Rijeka, Croatia
tihana.galinac@riteh.hr

³ Karlstad University, Karlstad, Sweden
andreas.kassler@kau.se, antonio.marotta@live.it

⁴ INESC-ID Lisboa/Instituto Superior Técnico,
Universidade de Lisboa, Lisbon, Portugal
pradeeban.kathiravelu@tecnico.ulisboa.pt, luis.veiga@inesc-id.pt

Abstract. Traditional networks are transformed to enable full integration of heterogeneous hardware and software functions, that are configured at runtime, with minimal time to market, and are provided to their end users on “as a service” principle. Therefore, a countless number of possibilities for further innovation and exploitation opens up. Network Function Virtualization (NFV) and Software-Defined Networking (SDN) are two key enablers for such a new flexible, scalable, and service-oriented network architecture. This chapter provides an overview of QoS-aware strategies that can be used over the levels of the network abstraction aiming to fully exploit the new network opportunities. Specifically, we present three use cases of integrating SDN and NFV with QoS-aware service composition, ranging from the energy efficient placement of virtual network functions inside modern data centers, to the deployment of data stream processing applications using SDN to control the network paths, to exploiting SDN for context-aware service compositions.

1 Introduction

Software-Defined Networking (SDN) is a new paradigm that provides programmability in configuring network resources. It introduces an abstraction layer on the network control layer that allows runtime and ad-hoc network reconfiguration. Therefore, it enables to adapt at runtime not only physical network resources but also software services that compose complex services delivered to end users. Such a new network feature thus provides a valuable mechanism to be exploited in the modeling of QoS-aware service compositions integrating services from various networks. This paradigm has been successfully incorporated into the virtualization of the telecommunication network and an architecture concept

called Network Function Virtualization (NFV), where virtual network functions are interconnected into service compositions to create communication services.

Traditional networks that have been designed for yesterday peak requirements are inefficient to cope with nowadays massive communication traffic injected by a large number of users (e.g., billions of devices in the Internet of Things). The main obstacle of traditional networks to provide full exploitation of their resources and accelerate innovation is caused by the lack of integration of the variety of hardware and software appliances. Moreover, the lack of standardized interfaces make network management costly and slow adapting to modern trends, and user demands [14, 20, 27].

Within the 5G network, SDN and NFV are the two key technologies introduced as enablers [33]. In future networks, the optimal cost is achieved through dynamic and self-adaptive deployment on a network infrastructure which is continuously controlling its performances and autonomously managing its resources. The primary goal of such a dynamic and autonomous deployment is to accomplish and maintain the quality of service (QoS) requirements of complex services. By adopting SDN and NFV for the composition of complex services, Software-Defined Service Composition (SDSC) [21] separates the execution of service compositions from the data plane of the overall system.

SDSC facilitates the integration and interoperability of more diverse implementations and adaptations of the services. A reliable execution of service composition can be guaranteed through the network management capabilities offered by SDN, in finding the best alternative among various service implementations and deployments among the multiple potential services deployments for the service composition execution. SDSC thus offers an increased control over the underlying network, while supporting the execution from various traditional web service engines and distributed frameworks.

There are various modeling approaches for QoS-aware service composition which have been proposed so far. With the introduction of a programmable approach to implement and use network resources, we should investigate performance modeling approaches that jointly consider all network layers and their composite behavior and outputs. Therefore, the contribution of this chapter is to analyze the integration of SDN and NFV in modeling the performance of service compositions and investigate possible side effects that can arise from their composite interactions. To this end, we present three different use cases of integrating SDN and NFV with QoS-aware service composition, ranging from the energy efficient placement of virtual network functions inside modern data centers, to the deployment of data stream processing (DSP) applications using SDN to control the network paths, to exploiting SDN for context-aware service compositions.

In the upcoming sections of this chapter, we continue to discuss the benefits and use cases of integrating SDN and NFV with QoS-aware service composition. Section 2 provides an overview of the basic concepts: SDN, NFV, and service compositions. Section 3 discusses the energy-efficient green strategies enabled by the integration of SDN and NFV with service compositions. Section 4 focuses

on a specific example of composite service - represented by DSP applications - and elaborates on the integration of a DSP framework with an SDN controller, showing a full vertical integration of the application and network layers. Section 5 discusses how SDN can offer context-aware service compositions. Finally, we discuss the benefits and open research issues in QoS-aware service compositions in Sect. 6 and conclude the chapter by identifying future research directions in Sect. 7.

2 Overview of Basic Concepts

A traditional network architecture divides Telco/Network operators from Internet Service Providers (ISPs) and Content Providers. Services are provided over highly specialized technologies which limit their full exploitation by end users. A new network architecture that is proposed for future networks introduces new abstraction layers with standardized interfaces that would enable Telco/Network Providers, ISPs, and Content Providers to provide their services over the web, independently from the underlying network. The vision of future networks is to provide their users with complex services that result from the autonomous composition of simple, possibly legacy, elementary services. Such a service orientation has also been recently reaffirmed for the next decade in the Service Computing manifesto [6], that call for the widespread adoption of service computing.

2.1 Introduction to NFV

The basic concept of NFV is to apply Cloud computing technologies to realize telecommunication applications. NFV revolves around the concept of virtualization, which enables to run multiple systems in isolation on a single hardware system. The exploitation of virtualization allows to decouple network functions from the related (dedicated) hardware [17]. In other words, a software implementation of different network functions (e.g., modulation, coding, multiple access, firewall, deep packet inspection, evolved packet core components) can be deployed on top of a so-called hypervisor, which runs on commercial off-the-shelf servers instead of dedicated hardware equipment. The hypervisor provides for virtualization and resource management (e.g., scheduling access to CPU, memory, and disk for the network functions). In addition, an orchestration framework needs to be in place, so to combine different virtual functions to obtain higher layer service chains implementing the end-to-end service. Moreover, the orchestration framework manages the deployment (e.g., which virtual function to place on what physical server) and the life cycle of the virtual network functions, including the management of their scalability. The latter comprises several tasks, among which monitoring performance, scaling either vertically or horizontally resources (i.e., either acquiring more powerful computing resources or spawning more replicas of the same virtual network function and load balancing among them).

Consequently, Virtual Network Functions (VNFs) are different from classical server virtualization technologies because VNF may form service chains composed of multiple virtual network functions, that exchange traffic which may be

deployed on one or multiple virtual machines running different network functions and replacing thus a variety of hardware appliances [33]. Such software implementation of network functions is easily portable among different vendors and may coexist with hardware-based platforms. Thus, the main benefits provided are a reduction of capital and operational expenditures, offering a reduced time-to-market as well as scalability to different resource demands.

However, with the introduction of VNFs, additional problems may arise, such as increased complexity. Additional interfaces need to be defined and maintained (e.g., between the hypervisor and the orchestration system), which leads to more complex system design. In addition, as applications can have strict requirements in terms of latency, performance guarantees are more difficult to be satisfied. This is because a given implementation of a VNF may perform differently when deployed on different hardware. For example, the deployment of I/O intensive VNF (e.g., a home subscriber service) on a server equipped with a standard HDD may lead to lower performance than the one resulting from a deployment on a server equipped with an SSD or NV-RAM. Consequently, new benchmarking tools are required that allow correlating the performance of a given VNF when deployed on a given hardware with a certain configuration.

2.2 Introduction to Service Composition Using SDN

The second enabling technology is SDN, which separates the network control plane from the infrastructure (data) plane [31]. It involves logical centralization of network intelligence and introduces abstraction of physical networks from the applications and services via standardized interfaces. SDN is considered an enabling technology for high volumes of traffic flows and responds “at runtime” on dynamic demand for network resources by avoiding time-consuming and costly manual reconfiguration of the network. Thus, it increases network resource exploitation and decreases time to market. Furthermore, service-orientation is introduced to enable the runtime discovery and deployment of services. When combined with NFV and SDN technologies, this feature can significantly improve the efficiency of network operations.

Figure 1 presents a high-level architecture, emphasizing three distinct management layers that are coordinated by a vertical deployment manager to provide possibly coordinated QoS-aware decisions about service deployment.

At the *infrastructure layer*, routers and switches are distributed over the network topology. These devices have their logical representation that is used for control and management purposes. Decisions of centralized network control are transferred over the standardized physical interfaces to operate over devices in this layer.

Network resources are virtualized in the *virtualization layer*. Each virtual resource has its logical representation that enables efficient management. The virtual resources may be interconnected into a graph-like topology. Again, autonomous decisions about their interconnection and placement are subject of the management entity at this layer.

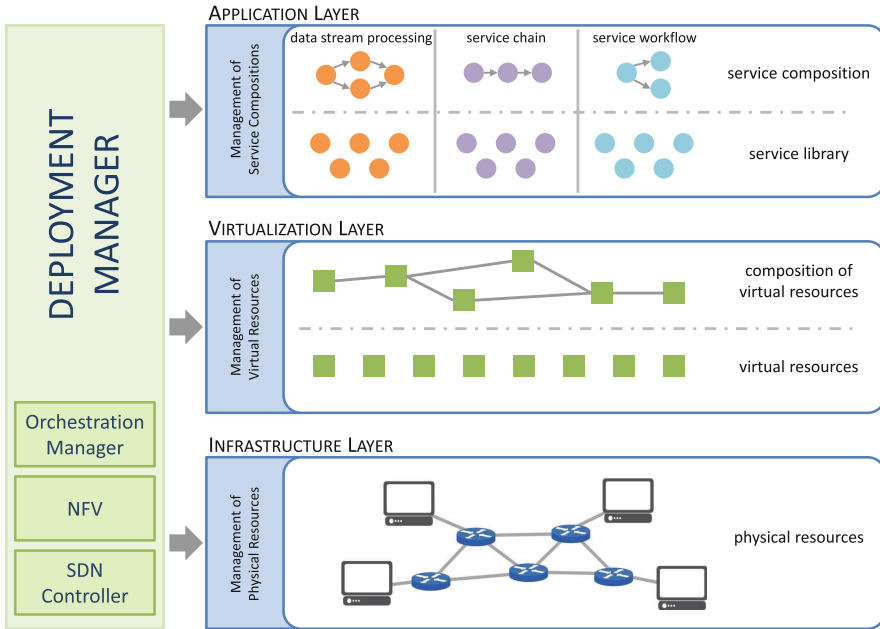


Fig. 1. High-level network overview.

Finally, at the *application layer* a number of basic component services are available in distributed data centers and exposed in service libraries. The complex services may be composed of many basic services that are accessible through service registries and can be composed on the basis of different goals. In the three use cases, we present later in this chapter, we consider network service chains, Web service and eScience workflows, and data stream processing (DSP) applications. A network service chain allows assembling services out of multiple service functions typically using basic patterns for service composition, e.g., a sequence of VNFs, with one or multiple instances needed for each VNF. Web services and eScience workflows usually organize their component services using more complex workflow patterns, e.g., conditional choice, loops, and fork-and-join. Finally, a DSP application is represented as a directed acyclic graph, that can be seen as a workflow diagram.

A service composition deployment on top of SDN allows cross-layer optimizations, as the services interact with the SDN controller through its northbound Application Programming Interface (API) protocols and using REpresentational State Transfer (REST) [39], Advanced Message Queuing Protocol (AMQP) [42], and Message Queue Telemetry Transport (MQTT) [32] transport protocols. On the other hand, the SDN controller orchestrates the data center network that the services are deployed on, through its southbound API protocols such as OpenFlow [28] and NetConf [16]. Such a cross-layer optimization supported by SDN allows QoS guarantees at the service and network levels.

NFV and SDN do not rely on each other. NFV is providing flexible infrastructure, while SDN software can run and can provide flow-based configuration of network functions. Both technologies, when used in cooperation, can offer enhanced QoS guarantees. In such new network architecture, the network logic is abstracted on several layers of abstraction. The management decisions of each layer may have reflections on the QoS provided by the network. Thus, the selection of collected management decisions within *deployment manager* should balance between flexibility provided at each level of network abstraction and optimal QoS.

An ongoing standardization endeavor is Next Generation Service Overlay Network (NGSON), aiming to establish a collaborative framework among the stakeholders from various networks and technology paradigms in order to unify their vision on common service delivery platform. Thus, the end-user need for complex service delivery across the network borders would be satisfied. The standard aims to identify self-organizing management capabilities of NGSON including self-configuration, self-recovery, and self-optimization of NGSON nodes and functional entities.

2.3 Overview of Use Cases

In this chapter, we will look into three illustrative use cases of integrating SDN and NFV with QoS-aware service composition.

Section 3 presents an overview on green strategies for VNF embedding, supported by SDN and NFV. Here, the key idea is to manage the NFV infrastructure, namely the composition of compute and networking resources including servers and networking equipment in an energy efficient way. By powering down unused servers and switches, the total energy of the infrastructure can be minimized. Important questions to ask are then what is the minimum number of servers, switches, and links that are necessary in order to provide the SLA desired for the service chains that need to be embedded into the physical network and compute infrastructure, where to place the functions and how to route the service chain traffic in order to find a balance between energy efficiency, performance and SLA.

Section 4 presents how the integration of an SDN controller with a DSP framework allows to adjust the network paths as per-application needs in the QoS-aware deployment of DSP applications on the computing and network resources. In the proposed integrated framework, SDN is used to expose to the DSP framework the network topology and network-related QoS metrics. Such information is exploited in a general formulation of the optimal placement problem for DSP applications, which jointly addresses the selection of computing nodes and of network paths between each pair of selected computing nodes.

We define services that access, process, and manage Big Data as big services. They pose computation and communication challenges due to their complexity, volume, variety, and velocity of Big Data they deal with. Moreover, they are often deadline-bound and mission-critical. Each big service is composed

of multiple services to be able to execute it in the Internet-scale at the distributed clouds. Such a componentization of big service improves its resilience and latency-awareness. For example, consider a big service for weather forecast. It consists of various services including sensor data retrieval, data analysis services, and prediction. These component services are inherently distributed, including the ones that manage the actuators and the sensors in land, sea, and satellites. By leveraging the SDN and NFV paradigms, SDSC ensures an efficient service composition from the replicated and globally distributed services. Section 5 discusses how SDSC leverages SDN to build and efficiently execute complex scientific workflows and business processes as service compositions.

3 Green Strategies for VNF Embedding

Next generation 5G networks will rely on distributed virtualized datacenters to host virtualized network functions on commodity servers. Such NFV will lead to significant savings in terms of infrastructure cost and reduced management complexity. Virtualization inside modern datacenters is a key enabler for resources consolidation, leading towards green strategies to manage both compute and network infrastructures where VNFs are hosted. However, green strategies for networking and computing inside data centers, such as server consolidation or energy aware flow routing, should not negatively impact on the quality and service level agreements expected from network operators, given that enough resources exist. For example, given two different resource allocation strategies, one focusing on performance while the other focusing on energy efficiency, while both strategies may lead to a resource allocation that satisfies user demands and SLAs, a green strategy does so by minimizing the energy consumption. Once fewer resources are available than requested, green strategies should guide the resource allocation processes towards operational points that are more energy friendly.

Important tools available for Cloud Operators are server consolidation strategies that migrate Virtual Machines (VMs) towards the fewest number of servers and power down unused ones to save energy. As VNFs are composed of a set of VNF Components (VNFC) that need to exchange data over the network under capacity and latency constraints, the networking also plays an important part. By using SDN, one can dynamically adjust the network topology and available capacity by powering down unused switch ports or routers that are not needed to carry a certain traffic volume [19], thus consuming the least amount of energy at a potential expense of higher latency. Green strategies try to place the VNFC onto the fewest amount of servers and to adjust the network topology and capacity to match the demands of the VNFCs while consuming the least amount of energy for operating the VNF Infrastructure. Such design of the VNF placement and virtual network embedding can be formulated as a mathematical optimization problem, and efficient heuristics can be designed to quickly solve the problem.

We can consider the Virtualized Compute and Network Infrastructure as the set of hardware resources (which is comprised of the compute and network

infrastructure) that is hosting a certain number of VNFs inside a virtualized data center. The virtualized data center can be geo-distributed to serve different users at different locations using the lowest cost in terms of energy, network, etc. We assume that each VNF is made of a set of service chains, which is a group of VNFC which have a set of traffic demands and a maximum tolerable latency allocated towards them. More precisely, the traffic demands specify how much traffic, between two adjacent services in a chain, the first sends to the second one. A service needs resources, e.g., in terms of CPU, memory, disk, and so on, to process packets and then forward the processing results to the next component of the chain.

The latency of a service chain is the sum of the experienced delays on the used paths, on which all the demands of the service chain are forwarded. It also includes the host internal processing related latency, which may be different for different architectural setups. For example, using standard Linux networking approach leads to much higher latency and less available capacity compared to using the recently developed approaches for user-mode packet forwarding and processing based on proprietary techniques, such as Intel's Data Plane Development Kit (DPDK).¹ Similarly, Single Root Input/Output Virtualization (SR-IOV²) is an extension to the PCI-express standard that allows different virtual machines (VMs) hosting the VNFs in a virtual environment to share a single network card over fast PCI-express lanes. Consequently, the additional latency for VNF packet processing depends on the virtualization technology used in the servers, which may be different for different server types. In addition, when two VNFC are placed on the same server, there is also a not negligible overhead when forwarding the packets from one component to another (after proper processing) and this overhead (and thus the additional latency and capacity limits) also depends on the virtualization technology used.

In the following, we assume that we have available a set J of servers and a network graph $G(N, E)$, where N represents the set of network nodes and E denotes the links among them. Given the family of service chains, which are defined as a specific number of traffic demands between couples of a subset $\bar{V} \subset V$ out of all VNFC, the objective of the problem is to allocate all the VNFCs on the servers and to find the network routes that satisfy the traffic demands while minimizing the overall power consumption P_{VNI} of the Virtual Network Infrastructure, which is the sum of the power consumption of the compute (P_{servers}) and network infrastructure (P_{switches}), given the latency, resource and bandwidth capacity budgets:

$$\min f = P_{VNI} = P_{\text{servers}} + P_{\text{switches}} \quad (1)$$

The key idea for developing green strategies is to place the network functions on the minimum number of servers and use the minimum number of highly energy

¹ <https://software.intel.com/en-us/networking/dpdk>.

² <https://www.intel.com/content/dam/doc/white-paper/pci-sig-single-root-io-virtualization-support-in-virtualization-technology-for-connectivity-paper.pdf>.

efficient network nodes that can serve the required capacity. Consequently, all unused servers and switches can be powered down to reduce energy consumption.

3.1 Power Model Examples for Compute and Network Infrastructure

Several power models have been proposed for the compute infrastructure. Typically, they assume that the CPU of a server is the most power hungry component [35], and consequently most models just consider the power consumption due to CPU load. In general, the relationship between server power consumption and CPU utilization is linear [24,36] with some small deviations that are due to processor architecture, CPU cache related aspects and compiler optimizations leading to a different CPU execution. For performance modeling of green server and network consolidation strategies, we can simplify that for each server j there is a unique idle power consumption $P_{idle,j}$, which denotes the energy required by the server when it is just powered on and does not run any compute (except the basic Operating System and management services). The maximum power consumption $P_{max,j}$ denotes the power consumed by the given server when all the CPU cores are under full load. In between the two extreme cases, the power consumption follows a linear model dependent on the CPU utilization.

The network related power consumption can also be simplified to make it tractable in numerical models. For example, the work in [5] assumes that for network switches there are two main components that impact the total power consumption. A static and constant power is required to power the chassis and the line cards, which is independent of the traffic that the switch serves and the number of ports used. In addition, depending on the number of ports per line rate are powered on, there is a dynamic power consumption, which also depends on the link speed the port is using (e.g., 1 Gbps or 10 Gbps) and the dynamic utilization of the ports. The power consumption also depends on the switch manufacturers: the work by Heller et al. [19] provides an overview on the power consumption of three different 48-port switch models. For example, one switch has a power consumption of 151 W when the switch is idle and all the ports are powered down, while it increases to 184 W when all the ports are enabled and to 195 W when all the ports serve traffic at 1 Gbps. As one can see, just powering on a switch requires the highest amount of power, while powering on additional ports does not add much to the total power consumption while the traffic dependent power consumption is almost negligible. Consequently, many green strategies try to conserve energy by powering down unused switches and power down unused ports.

3.2 Illustrative Example

In this section, we provide a simple example to illustrate the problem in Fig. 2. We assume there are seven servers (labeled from s_1 to s_7), each one with its own dedicated power profile specified by a given idle power P_s^{min} and maximum power consumption P_s^{max} . Each server has limited resources in terms of,

e.g., CPU, memory and disk capacities. To be more specific, server s_i has available a_{1i} CPU, a_{2i} RAM and a_{3i} DISK. Each server is connected to a specific router (e.g., the Top of Rack Switch in case of a Data Center).

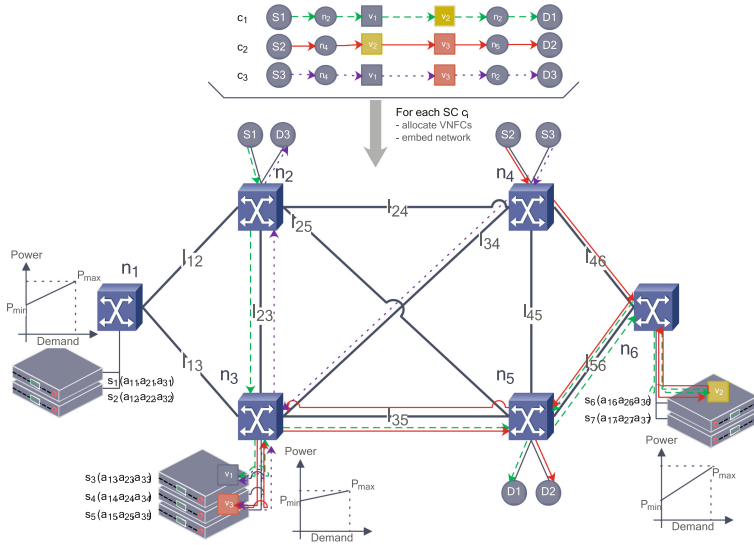


Fig. 2. The joint VNF placement and network embedding problem [26].

Each link that connects the servers to the switch or the switches with each other has a dedicated capacity and latency. In the example, the latency for the link between n_1 and n_2 is denoted as l_{12} . The latency has typically several components. The first one is the latency due to the capacity that the links operate, which is constant. There is also latency due to the virtualization technique applied, which depends on the load of the servers and other configurations (e.g., CPU cache misses). Furthermore, there is a load-dependent latency due to queuing, which is typically non-linear. However, under low load, such latency can be assumed to be linearly increasing, while under higher load, we can use a piecewise approximation to model the latency due to traffic being routed over the interface. In addition, each link has a dedicated capacity (omitted from Fig. 2 due to complexity).

In the given example, we should embed into this NFV Infrastructure three service chains (c_1 , c_2 and c_3). Each service chain has its unique latency bound, a dedicated traffic source S_1 , S_2 and S_3 and sink D_1 , D_2 and D_3 . For example, in 5G for machine-to-machine traffic low latency should be enforced while for multimedia traffic latency bounds could be more relaxed. Also, the model can be specified flexibly to model also control plane related service chains, with more stringent delay requirements. In the example, we have three different VNFs (v_1 , v_2 and v_3) and we assume that the traffic source for c_1 is the Sender S_1 ,

which is connected to router n_2 and injects a certain volume of traffic into the service chain towards v_1 . Then, v_1 processes the packets (for which it needs resources such as CPU, memory, and disk) and forwards the processed traffic (which may have a different volume than the one injected) towards VNFC v_2 , which again processes it and forwards a certain volume to the destination D_1 that is connected to router n_2 .

Note that Fig. 2 assumes additional source/sink nodes where traffic for a service chain is created/terminated. The figure shows an example of joint VNF placement and network embedding into the physical substrate network. VNFC v_1 would be placed onto server s_3 , v_3 onto server s_4 , and so on. Servers hosting no VNFC would be powered down (s_1 , s_2 , s_5 , s_7) together with all the nodes not carrying any traffic (n_1).

4 Integrating SDN into the Optimal Deployment of DSP Applications

In the section, we present a use case of integrating SDN with QoS-aware service composition that focuses on Data Stream Processing (DSP) applications. The advent of the Big Data era and the diffusion of the Cloud computing paradigm have renewed the interest in DSP applications, which can continuously collect and process data generated by an increasing number of sensing devices, to timely extract valuable information. This emerging scenario pushes DSP systems to a whole new performance level. Strict QoS requirements, large volumes of data, and high production rate exacerbate the need for an efficient usage of the underlying infrastructure. The distinguishing feature of DSP applications is their ability to processing data on-the-fly (i.e., without storing them), moving them from an operator to the next one, before reaching the final consumers of the information. A DSP application can be regarded as a composition of services [1] with real-time processing issues to address. It is usually modeled as a directed acyclic graph (DAG), where the vertexes represent the processing components (called application *operators*, e.g., correlation, aggregation, or filtering) and the edges represent the logical links between operators, through which the data streams flow.

To date, DSP applications are typically deployed on large-scale and centralized (Cloud) data centers that are often distant from data sources [18]. However, as data increase in size, pushing them towards the Internet core could cause excessive stress on the network infrastructure and also introduce high delays. A solution to improve scalability and reduce network latency lies in taking advantage of the ever-increasing presence of near-edge/Fog computing resources [4] and decentralizing the DSP application, by moving the computation to the edges of the network close to data sources. Nevertheless, the use of a diffused infrastructure poses new challenges that include network and system heterogeneity, geographic distribution as well as non-negligible network latencies among distinct nodes processing different parts of a DSP application. In particular, this latter aspect could have a strong impact on DSP applications running in latency-sensitive domains.

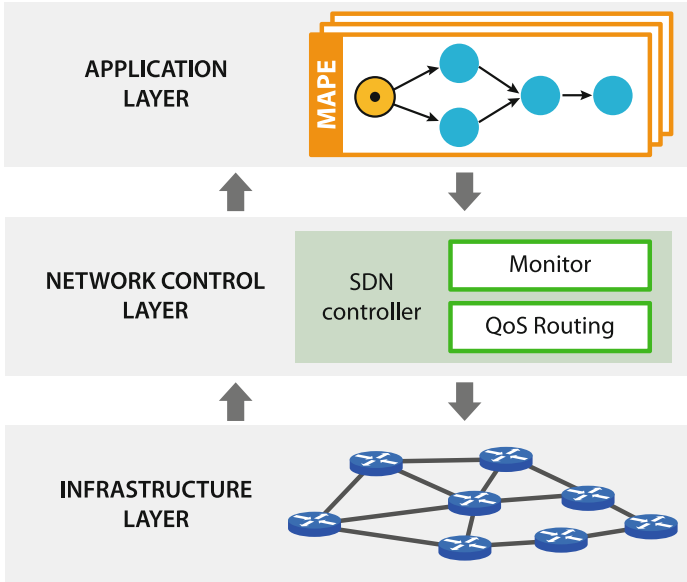


Fig. 3. DSP framework with SDN controller integration.

To address these challenges, we have proposed the solution depicted in Fig. 3 and named *SDN-integrated DSP Framework* (for short, SIDF), which combines and integrates a DSP application framework with an SDN controller. To this end, we have:

- extended the architecture of Apache Storm, a well known open-source DSP framework, by designing, developing, and integrating few key modules that enable a distributed QoS-aware scheduler architected according to the MAPE (Monitor, Analyze, Plan, and Execute) reference model for autonomic systems [7,8];
- designed, developed and implemented the controller logic for standard SDN controller and the associated API to provide network monitoring and dedicated stream routing configuration in an SDN network.

The proposed solution represents a full vertical integration of the application and network layers. The resulting architecture is highly modular and capable of taking full advantage of the SDN paradigm in modeling and optimizing the performance of Fog-based distributed DSP applications. In particular, SIDF enables the cross-layer optimization of the Fog/Cloud and SDN layers, whereby the SDN layer exposes to the upper layer the network topology and QoS metrics. This allows the optimal deployment of DSP applications by exploiting full knowledge of the computational and network resources availability and status. In this setting, an optimal deployment algorithm determines not only the application components placement on the underlying infrastructure but also the network paths between them.

For the sake of comparison with a non-SDN based solution, the proposed solution is backward compatible with legacy IP network, whereby network paths are solely determined by the underlying routing protocol and cannot be adjusted as per-application needs, thus providing no control by the DSP framework. In this setting, the DSP manager can at most monitor the network performance between candidate endpoints (see, e.g., [13] for a scalable network monitoring service) and determine operator placement on the underlying infrastructure by taking account the observed network delays.

4.1 The SIDF Architecture

SIDF uses a layered architecture to combine a DSP framework with an SDN controller (Fig. 3). The layered infrastructure enforces separation of concerns and allows to obtain a loosely coupled system. Each layer realizes new functionalities on top of lower-level services and exposes them as a service to the higher layer. SIDF comprises three main layers: infrastructure layer, network control layer, and application layer.

At the lowest level, the infrastructure layer and the network control layer represent the classical SDN network. Specifically, the *infrastructure layer* comprises network equipment, such as SDN devices and legacy IP devices. The former enables to monitor and dedicate communication paths, whereas the latter only exposes paths as black-boxes, resulting from their routing protocol.

The *network control layer* manages the heterogeneity of network devices and controls their working conditions. SIDF includes a network controller that realizes two functionalities: monitor and QoS routing. The monitoring components periodically observe the network so to extract metrics of interest; to limit the footprint of monitoring operations, we only retrieve network delays among network devices and computing nodes. Observe that these monitoring operations can be realized in an SDN controller assisted manner as proposed in [41], where the SDN controller periodically sends probes on links to measure their transferring delays, or in a distributed manner, where neighbor SDN devices autonomously compute latencies. As a result, the network control layer can expose a view of the infrastructure as a connected graph (or *network graph*), where network devices and computing nodes are interconnected by network links; the latter are labeled with monitoring information (e.g., network latency). Observe that, with legacy IP devices, the link between two network nodes represents the logical connectivity resulting from the routing protocols. As regards the QoS routing functionalities, the SDN controller allows installing dedicated stream routing configurations in the underlying infrastructure. Leveraging on the exposed network graph, the application layer of SIDF can instruct the network to route streams on specific paths, according to application needs. For example, the application might require to route data using either a best-effort path, the path that minimizes the number of hops, or the one that minimizes the end-to-end delay between two computing nodes.

The *application layer* includes the DSP framework, which abstracts the computing and network infrastructure and exposes to users simple APIs to execute DSP applications. Many DSP frameworks have been developed so far. Nevertheless, most of them have been designed to run in a clustered environment, where network delays are (almost) zero [9]. Since in an infrastructure with distributed computing resources (like in the Fog computing environment) network delays cannot be neglected, SIDF includes a custom distributed DSP framework that conveniently optimizes the execution of DSP applications. This framework, named Distributed Storm [8], has been implemented as an extension of Apache Storm [40], one of the mostly adopted open-source DSP frameworks. Distributed Storm oversees the deployment of DSP applications, which can be reconfigured at runtime so to satisfy QoS requirements (e.g., maximum application response time). To this end, the framework includes few key modules that realize the MAPE (Monitor, Analyze, Plan, and Execute) control cycle, which represents the reference model for autonomic systems [7, 8]. During the execution of MAPE phases, Distributed Storm cooperates with the other layers of SIDF so to jointly optimize the application deployment and the QoS-aware stream routing. Specifically, during the Monitor phase, the framework retrieves the resource and network conditions (e.g., utilization, delay) together with relevant application metrics (e.g., response time). Network conditions are exposed by the network control layer. During the Analyze phase, all the collected data are analyzed to determine whether a reconfiguration of the application deployment should be planned. If it is worth to reconfigure the application as to improve performance (or more generally, to satisfy QoS requirements), in the Plan and Execute phases the framework first plans and then executes the corresponding adaptation actions (e.g., relocate the application operators, change the replication degree of operators). The Plan phase determines the optimal deployment problem, whose general formulation is presented in the next section. If a reconfiguration involves changing the stream routing strategy, the Execute phase also interacts with the network control layer, so to enforce new forwarding rules.

4.2 DSP Deployment Problem

We now illustrate the optimal deployment problem for DSP applications with QoS requirements. We provide a general formulation of the optimal placement problem for DSP applications which jointly addresses the operator placement and the data stream routing by modeling both the computational and networking resources. A detailed description of the system model can be found in [9].

For a DSP application, solving the deployment problem consists in determining for each operator i :

1. the operator placement, that is the computational node where to deploy the operator i ;
2. the network paths that the data streams have to traverse from an operator i to each of the downstream operator j .

For the sake of simplicity, here we do not consider the operator replication problem, that is the determination of the number of parallel replicas for each operator to deploy in order to sustain the expected application workload. Nevertheless, the following arguments can be easily extended to the general case, e.g., using the approach presented in [10].

A deployment strategy can be modeled by associating to each operator i a vector $\mathbf{x}^i = (x_1^i, \dots, x_R^i)$, where $x_u^i = 1$, with $u \in \{1, \dots, R\}$ representing a computing resource, if the operator i is placed on the node u and 0 otherwise. Similarly, for each stream (i, j) from operator i to operator j , the vector $\mathbf{y}^{(i,j)} = (y_1^{(i,j)}, \dots, y_\Pi^{(i,j)})$, where $y_\pi^{(i,j)} = 1$, with $\pi \in \{1, \dots, \Pi\}$ representing a network path, if the data stream from operator i to operator j follows the path π_h and 0 otherwise.

The Operator Placement and Stream Routing (OPSR) problem takes the following general form:

$$\begin{aligned} \min \quad & F(\mathbf{x}, \mathbf{y}) \\ \text{subject to:} \quad & Q^\alpha(\mathbf{x}, \mathbf{y}) \leq Q_{\max}^\alpha \\ & Q^\beta(\mathbf{x}, \mathbf{y}) \geq Q_{\min}^\beta \\ & \mathbf{x}, \mathbf{y} \in A \end{aligned} \tag{2}$$

where $\mathbf{x} = (\mathbf{x}^{i_1}, \dots, \mathbf{x}^{i_n})$ is the vector of the operator deployment binary variables and $\mathbf{y} = (\mathbf{y}^{(i_1, j_1)}, \dots, \mathbf{y}^{(i_n, j_n)})$ is the vector of the network path variables.

Here, $F(\mathbf{x}, \mathbf{y})$ is a suitable objective function to be optimized which can conveniently represent application QoS metrics, e.g., response time, system and/or network related metrics, e.g., amount of resources, network traffic, or a combination thereof. $Q^\alpha(\mathbf{x}, \mathbf{y})$ and $Q^\beta(\mathbf{x}, \mathbf{y})$ are, respectively, those QoS attributes whose values are settled as a maximum and a minimum, and $\mathbf{x} \in A$ is a set of functional constraints (e.g., this latter set includes the constraint $\sum_u x_u^i = 1$, which requires that a correct placement deploys an operator on one and only one computing node, and $\sum_\pi y_\pi^{(i,j)} = 1$, which requires that, in a correct routing, a stream flows on a single path).

The formulation above represents the most general problem formulation whereby we jointly optimize the application deployment \mathbf{x} , by placing the operator on suitable nodes in the network, while at the same time determining the network paths \mathbf{y} to carry the stream between operators.

Using standard arguments, see, e.g., [9] for a similar problem, it can be proved that the resulting OPSR problem is NP-hard. As a consequence, efficient heuristics are required to deal with large problem instances in practice. Nevertheless, the proposed formulation can supply useful information for designing heuristics that, not only reduce the resolution time, but guarantee provable approximation bounds on the computed solution.

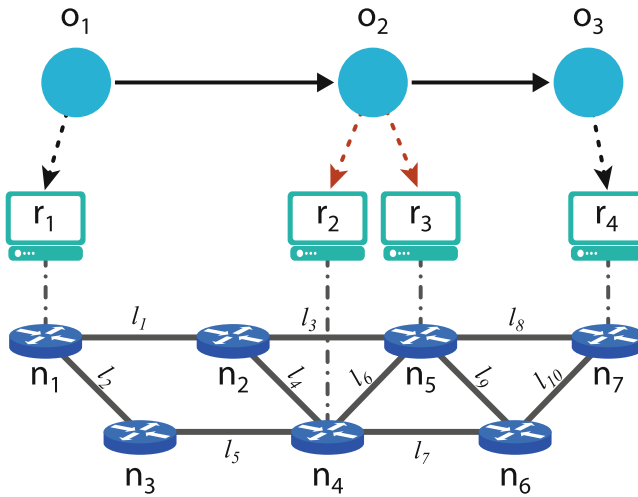


Fig. 4. SDN-supported placement of a DSP application.

4.3 Illustrative Example

OPSR determines how the computing and network resources should be utilized so to execute a DSP application with QoS requirements (e.g., response time, cost, availability). We observe that the application performance depends not only on computing resources, but also on network links that realize the communication among the computing nodes. This is especially true in geo-distributed environment (like Fog computing) and when Big Data have to be efficiently transmitted and processed. The strength of OPSR is the ability to jointly optimize (i.e., in a single stage) the selection of computing nodes and of network paths between each pair of selected computing nodes.

We exemplify the problem using Fig. 4. We consider a simple DSP application that filters and forwards important events to a notification service within a limited time interval (i.e., it has QoS requirements on response time). The application comprises a pipeline of three operators: a data source o_1 , a filter o_2 , and a connector to the external notification service o_3 . For the execution, OPSR has to identify computing and network resources from the available infrastructure that, in our example, comprises 4 processing nodes (r_i with $i \in \{1, \dots, 4\}$), 7 network devices (n_i with $i \in \{1, \dots, 7\}$), and 10 network links (l_i , with $i \in \{1, \dots, 10\}$ —observe that the network is not fully connected). To better show the problem at hand and reduce its complexity, we assume that each computing node r_i can host at most one operator and that o_1 and o_3 have been already placed on r_1 and r_4 , respectively. Therefore, OPSR has to deploy only the filtering operator o_2 selecting between two possible choices: r_2 and r_3 .

Interestingly, the network control layer can expose different views of the network, so that the upper application layer can select the most suitable network characteristics for running its applications. In our example, we consider that the

network control layer exposes paths with different QoS attributes in terms of communication latency and available bandwidth.

- In case o_2 is deployed on r_2 , OPSR has to further select the network paths for streams (o_1, o_2) and (o_2, o_3) , which should flow between (r_1, r_2) and (r_2, r_4) , respectively. For the first stream (o_1, o_2) , the network controller exposes $\pi_1 = \{l_1, l_4\}$, with 10 ms latency and 100 Mb/s bandwidth, and $\pi_2 = \{l_2, l_5\}$, with 25 ms latency and 1 Gb/s bandwidth. Similarly, for the second stream (o_2, o_3) , the network controller exposes $\pi_3 = \{l_6, l_8\}$, with 10 ms latency and 300 Mb/s bandwidth, and $\pi_4 = \{l_7, l_{10}\}$, with 15 ms latency and 850 Mb/s bandwidth.
- In case o_2 is deployed on r_3 , OPSR can determine the network paths for streams (o_1, o_2) and (o_2, o_3) , which should flow between (r_1, r_3) and (r_3, r_4) , respectively. For the first stream (o_1, o_2) , the network controller exposes $\pi_5 = \{l_1, l_3\}$, with 10 ms latency and 100 Mb/s bandwidth, and $\pi_6 = \{l_2, l_5, l_6\}$, with 30 ms latency and 600 Mb/s bandwidth. For the second stream (o_2, o_3) , the network controller exposes $\pi_7 = \{l_8\}$, with 5 ms latency and 100 Mb/s bandwidth, and $\pi_8 = \{l_9, l_{10}\}$, with 15 ms latency and 600 Mb/s bandwidth.

The utilization of any of these paths is upon request, because the SDN controller has to allocate resources so to guarantee that QoS performance does not degrade over time (e.g., due to link over-utilization). Since selecting one path or another deeply changes the application performance, OPSR picks the most suitable one driven by the DSP application QoS requirements, which are captured by the objective function $F(\mathbf{x}, \mathbf{y})$. Our DSP application needs to forward event notifications with bounds on delay, therefore it prefers to transfer data using the paths with minimum communication latency. Hence, OPSR maps o_2 on r_3 and selects the paths π_5 and π_7 , which introduce a limited communication latency of 15 ms. Observe that, in case the DSP application aimed to optimize the amount available bandwidth (as in case of media streaming applications), OPSR would have mapped o_2 on r_2 and selected the paths π_2 and π_4 , which provide a bandwidth of 1 Gb/s and 850 Mb/s, respectively.

Although this is a toy example, it gives a flavor of the potentialities coming from the cooperation between SDN and distributed DSP applications. At the same time, the example shows the combinatorial nature of the OPSR problem, which calls for the development of new efficient heuristics.

4.4 Related Work on Big Data and SDN

With the renewed interest in DSP applications, in the last years many research works have focused on the placement and runtime reconfiguration of DSP applications (e.g., [2, 9, 10, 25, 45] and therein cited works). However, some of these works [2, 45] do only consider the deployment of the DSP application in a clustered and locally distributed environment. Moreover, to the best of our knowledge, none of them exploits the support for the flexible and fine-grained programmable network control offered by SDN.

Enlarging the focus to Big Data applications, of which DSP applications represent the real-time or near-real-time constituent, SDN is considered as a promising paradigm that can help to address issues that are prevailing with such a kind of applications [11, 37]. These issues comprise data processing and resource allocation in locally and geographically distributed data centers, including micro data centers in Fog and edge computing, data delivery to end users, a joint optimization that addresses the tight coupling between data movement and computation, and application scheduling and deployment.

So far, in the Big Data scenario, most works have leveraged SDN to optimize the communication-intensive phase of Hadoop MapReduce [15] by placing MapReduce tasks close to their data, thus reducing the amount of data that must be transferred and therefore the MapReduce job completion time [29, 38, 43, 44]. A first work that explores the tight integration of application and network control utilizing SDN has been presented by Wang et al. [43], which explores the idea of application-aware networking through the design of an SDN controller using a cross-layer approach that configures the network based on MapReduce job dynamics at runtime. The Pythia system proposed by Neves et al. [29] employs communication intent prediction for Hadoop and uses this predictive knowledge to optimize at runtime the network resource allocation. The Pythia network scheduling component computes an optimized allocation of flows to network paths and, similarly to the QoS routing in our SIDF architecture, maps the logical flow allocation to the physical topology and installs the proper sequence of forwarding rules on the network switches. Xiong et al. propose Cormorant [44], which is a Hadoop-based query processing system built on top of SDN, where MapReduce optimizes task schedules based on the network state provided by SDN and SDN guarantees the exact schedule to be executed. Specifically, SDN is exploited to provide the current snapshot of the network status and to install the network path having the best available bandwidth. Their experimental results show a 14–38% improvement in query execution time over a traditional approach that optimizes task and flow scheduling without SDN collaboration. Qin et al. in [38] propose a heuristic bandwidth-aware task scheduler that combines Hadoop with the bandwidth control capability offered by SDN with the goal to minimize the completion time of MapReduce jobs.

The integration of SDN into the control loop of self-adaptive applications has been studied by Beigi-Mohammadi et al. [3] with the goal of exploiting network programmability to meet application requirements. This is a new trend in the design of self-adaptive systems. We also explore it with the SIDF architecture: the integration of SDN allows us to adapt at runtime the stream routing so that the QoS requirements of the DSP application can still be guaranteed when network operating conditions change. Besides the SDN appealing features, the strict cooperation between adaptive systems and the SDN controller might easily become a scalability bottleneck. Indeed, SDN controller are often implemented as a single centralized entity, whereas adaptive systems can span over geographically distributed infrastructures. Further research investigations are needed to enable the exploitation of SDN features in a scalable manner.

5 Context-Aware Composition of Big Services

Big services are typically composed of smaller web services or microservices, each with multiple alternative deployments to ensure performance, scalability, and fault-tolerance. Such service compositions enable the design and implementation of complex business processes, eScience workflows, and Big data applications, by aggregating the services. Services are often implemented using several approaches, languages, and frameworks still offering the same API, standardized as RESTful or Service Oriented Architecture (SOA) [30] web services.

As the demand for QoS and data quality is on the rise, along with the ever-increasing scale of Big data, service compositions execute in computational nodes that are geographically distributed in the Internet-scale. SDN can be extended and leveraged to manage the underlying network that interconnects the building blocks of such complex workflows, to enhance the scalability and potential use cases in services computing. An integration of SDN and NFV into service composition facilitates efficient context-aware distribution of service execution closer to the data, minimizing latency and communication overhead.

5.1 Software-Defined Service Composition (SDSC)

SDSC is an approach to a distributed and decentralized service composition, which leverages SDN for an efficient service placement on the service nodes. Following the SDSC approach, a typical eScience workflow is mapped onto a geographically distributed service composition. SDSC exploits both the data-as-a-service layer and network layer for the resource allocation. System administrators can monitor the health of the service compositions, through the web service engines that host the services, by observing the runtime parameters such as the executed requests and the requests on the fly can be monitored. The list of multiple web service deployments can be retrieved from the web service registry. In addition to these, SDSC leverages the global network knowledge of the SDN controller to find the network parameters such as bandwidth utilization to fine tune the services placement, offering features such as congestion control and load balancing, which can better be achieved in the network layer.

By separating the execution from the data plane of the overall system, SDSC facilitates integration and interoperability of more diverse implementations and adaptations of the services. A resilient execution of service composition can be guaranteed through the network management capabilities offered by SDN, in finding the best alternative among various service implementations and deployments among the multiple potential services deployments for the service composition execution. SDSC thus facilitates an increased control over the underlying network, while supporting the execution from various traditional web services engines and the distributed execution frameworks.

The core of SDSC is constituted by the communication between inter-domain SDN controllers, facilitated by various Message-Oriented Middleware (MOM) [12] protocols such as AMQP and MQTT. The service requests are mapped to the network through SDN, and the resource provisioning is managed

with the assistance of the SDN controller. Hence, each domain is aware of the services that are served by the services hosted in them. By offering communication between inter-domain controllers, resources are allocated efficiently for each service request.

There is an increased demand for configurability to service composition. Context-aware service composition is enabled by exploiting SDN in deploying service compositions. The Next Generation Service Overlay Network (NGSON) specification offers context-aware service compositions by leveraging virtualization [20]. SDN and NFV support context-awareness and traffic engineering capabilities [34], to manage and compose services. Research efforts focus on efficient resource utilization as well as enabling pervasive services [23] motivated by the standardization effort of NGSON.

5.2 Componentizing Data-Centric Big Services on the Internet

Workflows of mission-critical applications consist of redundancy in links and alternative implementations and deployments in place, either due to parallel independent developments or developed such to handle failures, congestion, and overload in the nodes. Distributed cloud computing and volunteer computing are two examples that permit multi-tenant computation-intensive complex workflows to be executed in parallel, leveraging distributed resources.

Figure 5 represents a multi-tenant cloud environment with various tenants. The tenants execute several big services. Many aspects such as locality of the executing cloud data center and policies must be considered for an efficient execution of the service workflow. An SDN controller deployment can ensure QoS to the cloud, by facilitating an efficient management of the network-as-a-service consisting of SDN switches, middleboxes, and hosts or servers. The controller communicates with the cloud applications through its northbound API, while controlling the SDN switches through its southbound API. Thus, SDN facilitates an efficient execution of big services.

In practice, no complex big service is built and deployed as a singleton or a tightly coupled single cohesive unit. Mayan [21], which is a distributed execution model and framework for SDSC, defines the services that compose a big services workflow as the “building blocks” of the workflow. SDSC aims to extend the SDN-enabled service execution further to the Internet-scale.

Representation of the Model. We need to consider and analyze the potential execution alternatives of the services, to support a context-aware execution of service compositions. In this section, we formally model the big services as service compositions and consider the potential execution alternatives for their context-aware execution. Services are implemented by various developers following different programming languages and paradigms.

$\forall n \in \mathbb{Z}^+; \forall \alpha \in \{A, B, \dots, N\}: s_\alpha^n$ represents the α^{th} implementation of service s^n .

Each implementation of a service can have multiple deployments, distributed throughout the globe, either as replicated deployments or independent

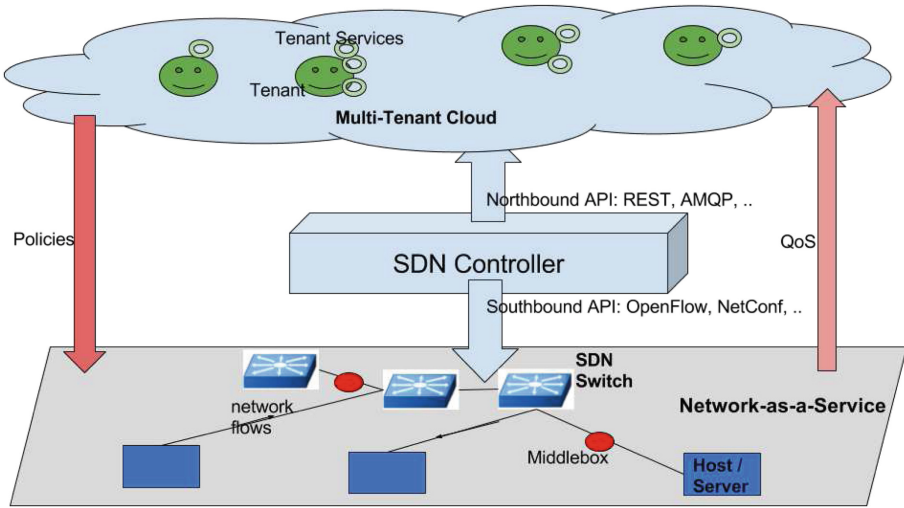


Fig. 5. Network- and service-level views of a multi-tenant cloud.

deployments by different edge data centers. These multiple deployments facilitate a bandwidth-efficient execution of the services.

$\forall m \in \mathbb{Z}^+ : s_{\alpha m}^n$ represents the m^{th} deployment of s_{α}^n .

Each service can be considered a function of a varying number of input parameters. Any given big service S can be represented as a composite function or a service composition. These service compositions are composed of a subset of globally available services.

$\forall x \in \mathbb{Z}^+, x \leq n; S = s^1 \circ s^2 \circ \dots \circ s^x$.

The minimum number of execution alternatives for any service can be represented by κ_x , where:

$$\forall s \in S : \kappa_x = \sum_{\alpha=A}^N m_{\alpha}$$

Here, N different service implementations and a varying number m_{α} of deployments for each implementation of s are considered.

Minimum and Maximum Execution Alternatives. Now we will formalize the maximum and minimum execution alternatives for any service composition, considering the multiple implementations or deployed replicas of the same service. More execution alternatives will offer more resilience and scalability to the service composition.

η_S represents the number of alternative execution paths for each big service S . The service that has the minimum alternatives limits the minimum number of potential alternatives for a service composition.

$$\eta_S \geq \min(\kappa_x : x \leq n) \geq 1.$$

Taking into account the alternatives due to various service combinations in the big service, the maximum alternatives is limited by a product of alternatives for each service.

$$\eta_S \leq \prod_{x=1}^n \kappa_x.$$

Hence,

$$\min(\kappa_x: x \leq n) \leq \eta_S \leq \prod_{x=1}^n \kappa_x.$$

Various protocols and web services standards unify the message passing between the services, and enable seamless migration among the alternatives, in a best-effort and best-fit strategy. SOA and RESTful web services support common message formats through standardizations. These efforts unify and revolutionize the way services are built on the Internet.

5.3 Illustrative Example

Figure 6 illustrates a sample workflow that represents a service composition. This workflow can be an eScience workflow or a complex business process. The workflow represents multiple possible execution paths when the service composition is decomposed or componentized into services (Services 1, 2, ..., n). A, B, C, ..., Z represents the alternative implementations for each of the services. Thus, service implementations such as 1A, 1B, and 1Z can function as an alternative to each other (here, each of these is an implementation of service 1).

As illustrated by Fig. 6, if service 3A is either congested or crashed, the service execution can be migrated to the next best-fit (chosen based on locality or some other policy) deployment 3B. (2,3)Z represents a service that is equal to the service composition of 3A(2A), the output of 2A as an input to 3A. Hence, it is not an alternative to 2A or 3A. It is also possible that not all the services have alternative deployments in considered environments (as indicated by the lack of Service 2 as in 2C). Service deployment details need to be specified in the service registry to be able to compose and execute the service workflows seamlessly.

5.4 eScience Workflows as Service Compositions

The Internet consists of various data-centric big services. Complex eScience workflows leverage multiple big services for their execution and can be decomposed into various geo-distributed web services and microservices. eScience workflows can, therefore, be represented by service compositions. Thus, these big services, centered around big data, can be expressed into simpler web services, which can be executed in a distributed manner.

Mayan seeks to find the best fit among the alternatives of available service execution options, considering various constraints of network and service level resource availability and requirements, while respecting the locality of the service requests. Mayan proposes a scalable and resilient execution approach to offer a multi-tenant distributed cloud computing platform to execute these services beyond data center scale.

Mayan enables an adaptive execution of scientific workflows through federated SDN controllers deployed in a wide area network. Hence, Mayan leverages

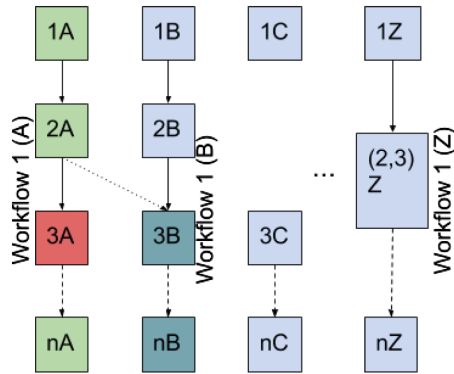


Fig. 6. Simple representation of multiple alternative workflow executions.

the various potential alternative execution paths existing between the service compositions, while exploiting the network knowledge of the SDN controller. Furthermore, Mayan utilizes the local workload information available at the web service engine and web services registry. The information received from this services layer includes web service requests on the fly and web services served at any time by the web service deployment. As an implementation of an SDSC, Mayan exploits both the control plane and services plane in offering a load-balanced, scalable, and resilient execution of service compositions. Mayan leverages OpenDaylight's data tree as an efficient control plane data store while using an AMQP-based messaging framework to communicate across multiple network domains in service resource allocation.

5.5 Inter-domain SDN Deployments

The SDN architecture needs to be extended for an Internet-wide service composition. A global view of the entire network hierarchy may not even be feasible to achieve for a single central controller due to the organizational policies. An inter-domain SDN deployment is necessary to cater for this scale and segregation of the network. Here, each domain (that can represent a cloud, organization, or a data center) is orchestrated by an SDN controller cluster.

The clustered deployment prevents the controller from becoming a single point of failure or a bottleneck. As eScience workflows are deployed on a global scale, a federated deployment of controller clusters is leveraged to enable communications between inter-domain controller clusters, without sharing a global network view. The federated deployment allows network level heuristics to be considered beyond data center scale, using MOM protocols in conjunction with SDN. Inter-domain controllers communicate through MOM messages between one another. Hence, SDN controllers of different domains have protected access to data orchestrated by one another, based on a subscription-based configuration rather than a static topology.

Some research work has previously leveraged federated SDN controller deployments for various use cases. CHIEF [22] presents a scalable inter-domain federated SDN controller deployment for wide area networks, as a “controller farm”. It builds a large-scale community cloud orchestrated by various independent controller clusters sharing data through a protected MOM API. Such controller farm may support collaboration between multiple organization networks, otherwise limited from network-level coordination. SDSC can be extended to create a Service Function Chaining (SFC), that is an ordered sequence of middlebox actions or VNFs such as load balancing and firewall.

6 Benefits and Open Issues

Network virtualization and programmability of network resources enable dynamic creation of service chains that satisfy QoS demands of complex services at runtime. Runtime control of traffic and usage of network resources is provided from infrastructure to control layer thus enabling runtime management decisions. Abstracting the network infrastructure plane is a movement similar as introducing higher levels of abstraction into programming languages. The key benefit of such abstraction is enabling less experienced developers to easier program new applications, using abstract objects of network resources, with the help of formal programming frameworks and environments. The risks of programmer faults are minimized through formalisms implemented in programming languages. The main benefit is in offloading new application developers of very complex network skills, thus opening application development even to not skilled people and innovation opportunities to the wider community. Abstraction of network resources will benefit with opening innovation opportunities based on the use of unlimited network resources.

A direct consequence of opening network resources to wider developers community is in accelerating the process of offering new features to end users and minimizing development costs. Another result of abstraction is the introduction of standard interfaces that enable evolution and change of each layer independently. Contrary to traditional networks where there is a dominant vendor lock-in solutions, in new network architecture, with introduced standard application platform interfaces between network layers, the independence to provider equipment has opened numerous opportunities for innovation by using an unlimited pool of network resources and services offered by various networks.

Furthermore, the programmable network enables numerous possibilities for network automation. New service management models may be developed at each network layer independently with runtime control of network resources. These may be used to autonomous control efficiency of network resource use while addressing specific QoS requirements of the particular application.

Nowadays, service compositions and Big Data applications must deal with changing environments and variable loads. Therefore, to guarantee acceptable performance, these applications require frequent reconfigurations, such as adjustments of application component placement or selection of new services. In this

respect, SDN capability of programming, the network at runtime allows a cross-layer management of computational and networking resources, thus enabling a joint optimization of application placement (or service composition) and data stream routing. The cross-layer management can be beneficial especially in geodistributed environments, where network resources are heterogeneous, subject to changing working conditions (e.g., congestion), and characterized by non-negligible communication delays. In an SDN environment, the application control layer (e.g., service composition broker, DSP framework) can regard the network as a logical resource, which can be managed as a computing resource in a virtualized computing environment. Specifically, the programmability allows to automate and control the network so to adjust its behavior as to fulfill the application needs. For example, multiple paths or paths with specific QoS attributes can be reserved for transmitting data, data streams can be redirected during application components downtime, or network devices can be programmed to carry out new functions. Moreover, the use of standardized interfaces between the application layer and network controller (i.e., Northbound APIs) allows simplifying the implementation and utilization of new network services (e.g., QoS-based routing).

With respect to the integration of SDN and Big Data and specifically to the SIFD architecture presented in Sect. 4, we observe that when the network controller in SDN is used for Big Data applications, its performance could be degraded due to the rapid and frequent flow table update requests which might not be sustained by today SDN controllers. The problem is exacerbated if the controller serves multiple applications/frameworks as it can easily become the performance bottleneck of the entire architecture. To this end, we need to define solutions which cater for the presence of multiple applications, with possible diverse and conflicting QoS requirements by defining policies which ensure fair usage of network resources in the face of competing resources requests. The problem becomes relevant in large-scale distributed environments, where a centralized approach might not scale, and distributed solution becomes preferable.

New service development formalisms may be required to standardize processes at the network management level. In the future use of such a programmable network environment, a network is seen as an unlimited pool of resources. So, it is expected a significantly increase in the network use with a number of new and innovative services. Such increase in diversity of network services and a number of new application interfaces would need to redefine service development and management models. New design principles would be needed, and this need would be recognized with increased diversity at network application layer. For such purposes, there is a need for new developments in formal methods for introducing the controlled behavior in programming network. Development of network compilers is ongoing research activity for these purposes. Furthermore, new mathematical models are needed that would be able to describe network behavior. There is a need for some generative models that can predict the parameters from the internal properties of the processes we are controlling. Such models would not only bring efficiency in processing network control algorithms, but would also be stimulating phenomena in network behavior.

7 Conclusions

In this chapter, we looked into how SDN and NFV enable QoS-aware service compositions, and how SDN can be leveraged to facilitate cross-layer optimizations between the various network and service layers. So far, SDN has been largely and separately exploited mainly in telecommunication environments. For example, NFV placement and SDN routing for network embedding have been used to achieve energy efficiency as explained in Sect. 3. However, there is an increasing interest in exploring the network control opportunities offered by SDN in the Big Data context, as discussed for the deployment of DSP applications on the underlying computing and networking resources. In the use case presented in Sect. 4, SDN is used to expose to the service management layer the network topology and network-related QoS metrics. The service management layer determines both the application components placement on the underlying computing resources and the network paths between them. In this way, SDN allows autonomous adjustment of the network paths as per-application needs. Furthermore, in Sect. 5 we provided an example of using SDN for the design and implementation of complex scientific and business processes.

Through these three examples, we presented different deployment management decisions for service compositions over the layers of a network architecture that integrates SDN and NFV. As future research direction, we identify the need for the development of an autonomous management framework that can coordinate cross-layer decisions taken by different management layers while deploying service compositions that satisfy QoS guarantees in an Internet-scale distributed network. Future work is also needed to investigate the side effects that may arise from the coordination among management decisions at different layers.

References

1. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., et al.: Aurora: a new model and architecture for data stream management. *VLDB J.* **12**(2), 120–139 (2003)
2. Aniello, L., Baldoni, R., Querzoni, L.: Adaptive online scheduling in Storm. In: *Proceedings of 7th ACM International Conference on Distributed Event-Based Systems, DEBS 2013*, pp. 207–218 (2013)
3. Beigi-Mohammadi, N., Khazaei, H., Shtern, M., Barna, C., Litoiu, M.: On efficiency and scalability of software-defined infrastructure for adaptive applications. In: *Proceedings of 2016 IEEE International Conference on Autonomic Computing, ICAC 2016*, pp. 25–34 (2016)
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: *Proceedings of 1st Workshop on Mobile Cloud Computing, MCC 2012*, pp. 13–16 (2012)
5. Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A.: Energy-efficient data replication in cloud computing datacenters. *Cluster Comput.* **18**(1), 385–402 (2015)
6. Bouguettaya, A., Singh, M., Huhns, M., Sheng, Q.Z., et al.: A service computing manifesto: the next 10 years. *Commun. ACM* **60**(4), 64–72 (2017)
7. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: On QoS-aware scheduling of data stream applications over fog computing infrastructures. In: *Proceedings of IEEE ISCC 2015*, pp. 271–276, July 2015

8. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Distributed QoS-aware scheduling in Storm. In: Proceedings of 9th ACM International Conference on Distributed Event-Based Systems, DEBS 2015, pp. 344–347 (2015)
9. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Optimal operator placement for distributed stream processing applications. In: Proceedings of 10th ACM International Conference on Distributed and Event-Based Systems, DEBS 2016, pp. 69–80 (2016)
10. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Optimal operator replication and placement for distributed stream processing systems. *ACM SIGMETRICS Perform. Eval. Rev.* **44**(4), 11–22 (2017)
11. Cui, L., Yu, F.R., Yan, Q.: When big data meets software-defined networking: SDN for big data and big data for SDN. *IEEE Netw.* **30**(1), 58–65 (2016)
12. Curry, E.: Message-oriented middleware. In: *Middleware for Communications*, pp. 1–28. Wiley, Hoboken (2005)
13. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.* **34**(4), 15–26 (2004)
14. Davy, S., Famaey, J., Serrat, J., Gorricho, J.L., Miron, A., Dramitinos, M., Neves, P.M., Latre, S., Goshen, E.: Challenges to support edge-as-a-service. *IEEE Commun.* **52**(1), 132–139 (2014)
15. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113 (2008)
16. Enns, R., Bjorklund, M., Bierman, A., Schönwälder, J.: Network Configuration Protocol (NETCONF). RFC 6241, June 2011
17. Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: challenges and opportunities for innovations. *IEEE Commun.* **53**(2), 90–97 (2015)
18. Heinze, T., Aniello, L., Querzoni, L., Jerzak, Z.: Cloud-based data stream processing. In: Proceedings of 8th ACM International Conference on Distributed Event-Based Systems, DEBS 2014, pp. 238–245 (2014)
19. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: ElasticTree: saving energy in data center networks. In: Proceedings of 7th USENIX Conference on Networked Systems Design and Implementation, NSDI 2010 (2010)
20. John, W., Pentikousis, K., Agapiou, G., Jacob, E., Kind, M., Manzalini, A., Risso, F., Staessens, D., Steinert, R., Meirosu, C.: Research directions in network service chaining. In: 2013 IEEE SDN for Future Networks and Services. SDN4FNS (2013)
21. Kathiravelu, P., Galinac Grbac, T., Veiga, L.: Building blocks of Mayan: Componentizing the escience workflows through software-defined service composition. In: Proceedings of 2016 IEEE International Conference on Web Services, ICWS 2016, pp. 372–379 (2016)
22. Kathiravelu, P., Veiga, L.: CHIEF: controller farm for clouds of software-defined community networks. In: Proceedings of 2016 IEEE International Conference on Cloud Engineering Workshop, IC2EW 2016 (2016)
23. Liao, J., Wang, J., Wu, B., Wu, W.: Toward a multiplane framework of NGSON: a required guideline to achieve pervasive services and efficient resource utilization. *IEEE Commun.* **50**(1) (2012)
24. Lim, S.H., Sharma, B., Nam, G., Kim, E.K., Das, C.R.: MDCSim: a multi-tier data center simulation platform. In: Proceedings of 2009 IEEE International Conference on Cluster Computing and Workshops, August 2009
25. Lohrmann, B., Janacik, P., Kao, O.: Elastic stream processing with latency guarantees. In: Proceedings of IEEE ICDCS 2015, pp. 399–410 (2015)

26. Marotta, A., D'Andreagiovanni, F., Kassler, A., Zola, E.: On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures. *Comput. Netw.* **125**, 64–75 (2017)
27. Matsubara, D., Egawa, T., Nishinaga, N., Kafle, V.P., Shin, M.K., Galis, A.: Toward future networks: a viewpoint from ITU-T. *IEEE Commun.* **51**(3), 112–118 (2013)
28. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **38**(2), 69–74 (2008)
29. Neves, M.V., De Rose, C.A.F., Katrinis, K., Franke, H.: Pythia: faster big data in motion through predictive software-defined network optimization at runtime. In: *Proceedings of IEEE 28th International Parallel and Distributed Processing Symposium, IPDPS 2014*, pp. 82–90 (2014)
30. Newcomer, E., Lomow, G.: *Understanding SOA with Web Services*. Addison-Wesley, Upper Saddle River (2005)
31. Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T.: A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun. Surv. Tutorials* **16**(3), 1617–1634 (2014)
32. OASIS: MQTT version 3.1.1 (2014)
33. Osseiran, A., Monserrat, J.F., Marsch, P.: *5G Mobile and Wireless Communications Technology*, 1st edn. Cambridge University Press, New York (2016)
34. Paganelli, F., Ulema, M., Martini, B.: Context-aware service composition and delivery in NGSONs over SDN. *IEEE Commun.* **52**(8), 97–105 (2014)
35. Panda, P.R., Silpa, B.V.N., Shrivastava, A., Gummidipudi, K.: *Power-Efficient System Design*, 1st edn. Springer, Boston (2010). <https://doi.org/10.1007/978-1-4419-6388-8>
36. Pedram, M., Hwang, I.: Power and performance modeling in a virtualized server system. In: *Proceedings of 39th International Conference on Parallel Processing Workshops, ICPPW 2010*, pp. 520–526 (2010)
37. Qadir, J., Ahad, N., Mushtaq, E., Bilal, M.: SDNs, clouds, and big data: new opportunities. In: *Proceedings of 12th International Conference on Frontiers of Information Technology*, pp. 28–33 (2014)
38. Qin, P., Dai, B., Huang, B., Xu, G.: Bandwidth-aware scheduling with SDN in Hadoop: a new trend for big data. *IEEE Syst. J.* **11**(4), 2337–2344 (2015)
39. Richardson, L., Ruby, S.: *RESTful Web Services*. O'Reilly Media, Inc., Sebastopol (2008)
40. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., et al.: Storm@Twitter. In: *Proceedings of ACM SIGMOD 2014*, pp. 147–156 (2014)
41. Van Adrichem, N.L., Doerr, C., Kuipers, F.A.: OpenNetMon: network monitoring in OpenFlow software-defined networks. In: *Proceedings of 2014 IEEE Network Operations and Management Symposium, NOMS 2014* (2014)
42. Vinoski, S.: Advanced message queuing protocol. *IEEE Internet Comput.* **10**(6) (2006)
43. Wang, G., Ng, T.E., Shaikh, A.: Programming your network at run-time for big data applications. In: *Proceedings of 1st Workshop on Hot Topics in Software Defined Networks, HotSDN 2012*, pp. 103–108. ACM (2012)
44. Xiong, P., He, X., Hacigumus, H., Shenoy, P.: Cormorant: running analytic queries on MapReduce with collaborative software-defined networking. In: *Proceedings of 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2015*, pp. 54–59 (2015)
45. Xu, J., Chen, Z., Tang, J., Su, S.: T-Storm: traffic-aware online scheduling in Storm. In: *Proceedings of IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014*, pp. 535–544 (2014)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

