Flexibility in Softwarized Networks: Classifications and Research Challenges

Mu He, Alberto Martínez Alba, Arsany Basta, Andreas Blenk, Wolfgang Kellerer
Technical University of Munich
Email: {mu.he, alberto.martinez-alba, arsany.basta, andreas.blenk, wolfgang.kellerer}@tum.de

***Abstract*—The increase of flexibility is a common objective of softwarized networks based on concepts such as Software Defined Networking, Network Function Virtualization and Network Virtualization. Hence, in the state of the art, flexibility is used as an argument for a certain proposed architecture, solution mechanism or design choice in general. The meaning of flexibility behind such rather qualitative arguments is highly diversified in the literature, as a common understanding of flexibility is missing so far. In this article, we survey the state of the art in softwarized networks with a focus on the flexibility that is provided by each proposed concept, mechanism or system. In particular, we show that the flexibility provided by different network softwarization technologies can be classified into six different aspects within the three high-level flexibility categories, i.e., configuration adaptation, functions location and scalability. We analyze the state of the art in flexibility from several viewpoints including flexibility aspects, network technologies, domains and planes in order to derive a common understanding of how flexibility can be provided in softwarized wireline and wireless networks. Moreover, we reveal open issues, which are mostly related to the fact that flexibility is not clearly defined in the literature, and derive concrete research challenges accordingly. Our classification of flexibility and the derived research challenges aim at stimulating the discussion towards a more quantitative analysis of the design requirement of flexibility that has demonstrated increasing importance for softwarized networks and beyond.**

I. Introduction

In recent years, flexibility has emerged as a key property of communication network designs to react to dynamic changes that reside potentially in user demands, traffic conditions and application requirements [1]. A recent survey on 5G technology [2], for instance, reports "flexible and scalable network" as the top motivation for technology investment of 297 companies. Furthermore, flexibility has become a key decision factor for network designs, as telecom companies seek to be able to face the uncertain future. In fact, according to [3], regulatory changes and fast arrival of new technologies are major concerns for 80% and 72% of these companies, respectively.

To support network adaptation and hence flexibility, the way we operate networks has become more software-oriented. Those *softwarized networks* rely on concepts such as Software Defined Networking (SDN) [4], Network Functions Virtualization (NFV) [5], and Network Virtualization (NV) [6], which provide a new level of indirection as well as new interfaces for programming the control plane and setting up (virtual) network functions and networks on demand.

In order to demonstrate the possibilities that we have with softwarized networks, we refer to the scenario illustrated in Fig. 1. A big event in the city (lower left) demands for high quality connectivity to a service, which is realized as Virtual Network Function (VNF) in the upper right data center. To support the connectivity, an SDN-based controller configures the flows on runtime and steers the traffic towards the service (green line). With NV, the traffic of the event can acquire its own Virtual Network (VN) and have exclusive virtual resources on the nodes and the links. In other words, it is isolated from other VNs, such as the one hosting the traffic originating from the factory (upper left). When the upper right data center becomes overloaded, one of the running VNFs is migrated to the other data center. VNF migration requires not only reconfiguration of NFV in the functionality, but also reconfiguration of SDN in the network. Similarly, a link failure in the infrastructure triggers the migration of two virtual links (dashed blue line) to maintain connectivity of the factory. As demonstrated by this scenario, softwarized networks support the accommodation of more dynamic changes.

In a highly dynamic environment, network flexibility becomes not only desirable but sometimes critical. That is, flexibility is considered a crucial characteristic of any network, alongside other indicators such as performance and cost. Indeed, when facing different network designs that have similar latency, throughput, and physical resource consumption, we would opt for the one that implies more flexibility. The reason is simple: if new demands emerge in the future, network flexibility provides us with the ability to cover those demands without the need of jumping to a new system, which may imply inflated costs. Therefore, a detailed quantification of flexibility will assist us in making better decisions while building up our network.

However, arguments about flexibility concepts are diverse within both academia and industry. Due to the lack of common understanding, a meaningful analysis and comparison of network designs with respect to flexibility is hardly possible. Apart from that, the vast design space of softwarized networks makes the comparison potentially more challenging. To cope with these issues, we need a comprehensive analysis of flexibility in softwarized networks. This analysis can help us to clarify the essence of flexible communication networks, i.e., derive common characteristics from various proposals. It also

opens further challenges to fuel research efforts towards future flexible networks.

In order to perform a comprehensive analysis with respect to flexibility, a number of challenging questions have to be addressed: What dynamic changes have to be supported? In what time frame is a reaction expected? What network domains are targeted? Wireless access, wired backbone or data center? Which of the concepts of SDN, NFV and NV and what combination is in focus? Which type of adaptation is applied to the flows and network resources? And last but not least: at what cost comes the realization of a more flexible network design?

Before we can address these questions, we need a basic definition of network flexibility. Recall the scenario in Fig. 1, the network design would be considered flexible if the high quality traffic demand of the big event, data center overload, and physical link failure can be accommodated within a short time frame. The definition of network flexibility we apply for our survey is based on this understanding. We assert that a network design is flexible, if it can support new requests via adapting its resources if needed within a given time threshold [7]. "New requests" stand for changes in the requirements such as variation in the traffic demands [7] or a set of new design goals such as shorter latency budgets [8]. The time threshold depends on the type of requests and may be infinite to express the general potential of adaptation. Similarly, flexibility is also an important characteristic in other fields, such as economics [9], [10], management science [11]–[13] and software engineering [14]–[16]. We pay special attention to the consistency of the above definition with these fields to make sure that we perform the analysis upon a most common understanding.

Based on our definition, we address the above questions with a detailed survey of the state of the art, perform a classification towards the technical approaches, and then draw concrete conclusions from the applied classification. In particular, the main contributions of this survey are as follows:

- We give a tutorial on network softwarization with a focus on the applicable technologies;
- We derive flexibility aspects from the concepts of SDN, NFV and NV to guide the classification of network flexibility in our survey;
- We analyze the state of the art of softwarized networks with respect to network flexibility based on our classification from different points of view;
- We discuss common observations in a detailed lessons learnt section and derive open research challenges for the future.

The classification analysis together with the resulting observations serves as an overview of the current state of the art in softwarized networks with respect to flexibility. Moreover, it aims at providing a new structure in the scattered network flexibility discussion and at contributing to a better understanding of network flexibility as a design goal. For system designers, it should support a meaningful comparison between different network designs and foster decision making for network design options. For researchers, it should stimulate a detailed discussion and show new research directions to provide flexibility in communication networks.

Whereas the input material for this survey are publications in the fields of SDN, NFV and NV, its scope and classification criteria are fundamentally different from those of existing surveys on these technologies. In contrast to general surveys on SDN, NFV and NV [4], [5], [17]–[25], [25]–[40], [40]–[44], we provide a new perspective on those softwarized network technologies focusing on network flexibility as a new metric for classification and as a potentially new quality indicator. Typical state-of-the-art surveys classify the publications with respect to existing performance indicators (e.g., [4], [21], [38], [44]), methodologies (e.g, [17], [19], [37], [39]), use-cases (e.g., [4], [33], [41]), etc. We derive our own set of flexibility aspects that are shared among softwarized network technologies and do not rely on general technology aspects for classification. With this, we aim at shedding new light on the research in softwarized networks. More details on existing surveys on SDN, NFV and NV, are given at the end of Sec. II-A, II-B, and II-C, respectively.

The remainder of this survey is organized as follows. Section II introduces the technologies that contribute to network softwarization. In Section III we provide a definition of network flexibility, as well as derived flexibility aspects and incurred cost, also supported by flexibility studied in other research areas. As the main part of the survey, Section IV comprehensively analyzes the state of the art proposals according to the flexibility aspects. We classify the state of the art further with other viewpoints, namely network domains and planes in Section V. Section VI compiles the observations and related insights, and Section VII highlights future research challenges. The survey is concluded finally in Section VIII.

II. Technologies for network softwarization

Network softwarization is enabled by various novel technologies, namely SDN, NFV and NV. In this section, we present the concepts of the technologies and other important terminologies that will be used throughout this survey. We also provide Fig. 2 to conceptually illustrate the technologies.

A. Software Defined Networking

Concept. SDN is a networking paradigm that promises more flexibility in network deployment and management. The network control logic, i.e., the control plane, is decoupled from the network forwarding entities (such as routers and switches), i.e., the data plane. For legacy routing protocols, such as distance-vector based IGRP [45] and link-state based OSPF [46], each forwarding device (router) makes the forwarding decision on its own, based on its knowledge of the network. This knowledge is acquired with the help of control protocols, which define the exchange of packets containing control information. For SDN, the separation of control plane and data plane results in "dummy" forwarding devices that only

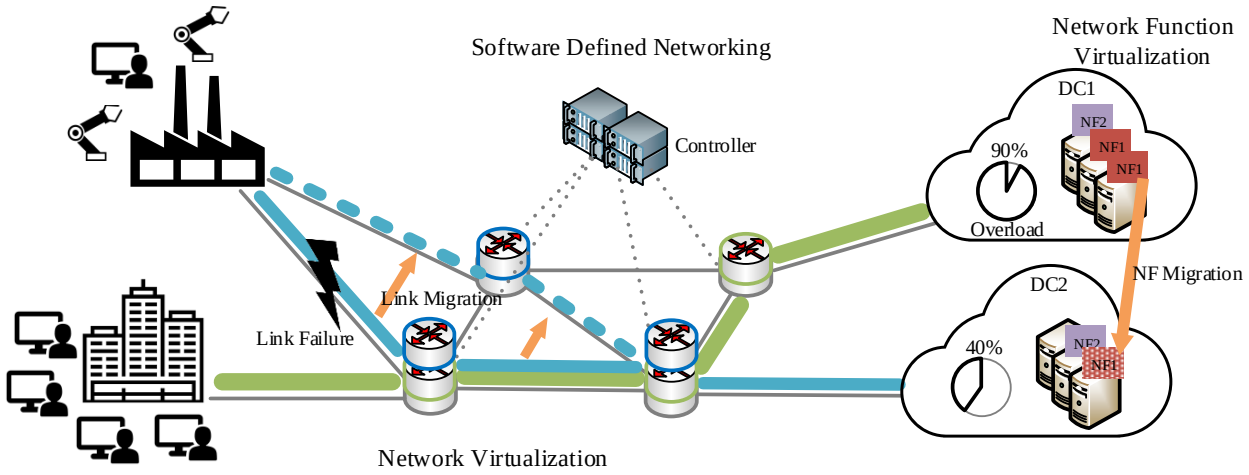


Fig. 1. Softwarized network scenario that combines technologies of SDN, NFV and NV. A central SDN controller directs traffic to particular network services upon demand, and sets up new path as physical link failure occurs. Networks serving various tenants are isolated and controlled dynamically in the same physical infrastructure. Network functions are virtualizedin different data centers. When one of the data centers is overloaded, some functions are migrated to the other data center to ensure load balancing.

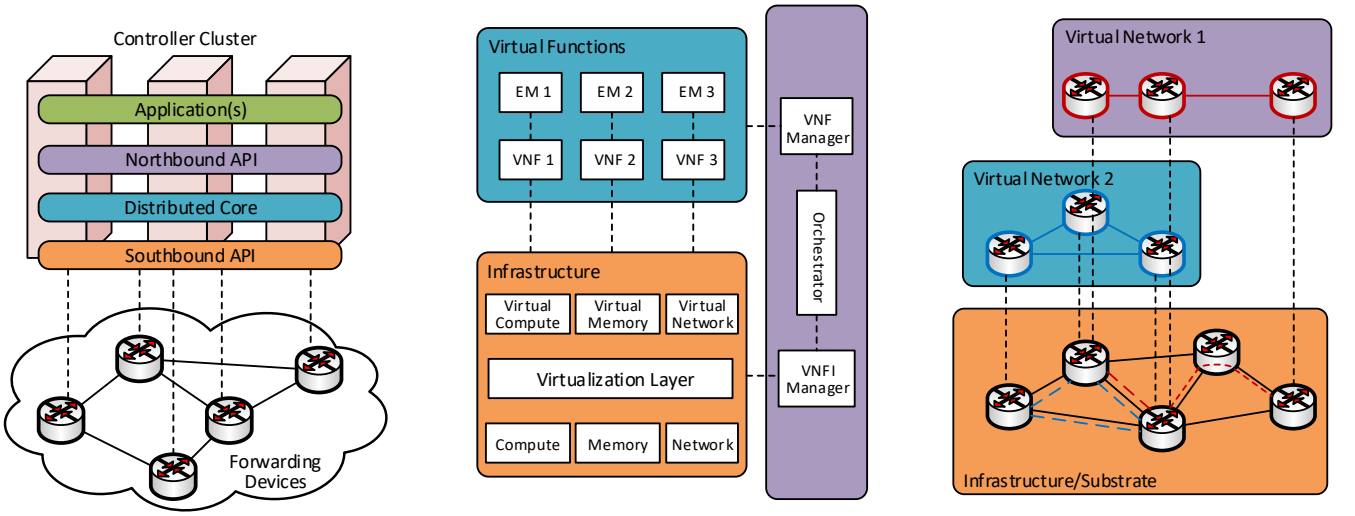
handle packet forwarding. The whole control logic is thereafter implemented as the SDN controller, in a logically centralized manner. Since the SDN controller has an overall view of the underlying network topology, it could make global optimal decisions in network traffic control, policy enforcement as well as system reconfiguration and upgrade [4], [47].

Control Interface. A programming interface is needed to inter-connect data plane and control plane. This interface, which is sometimes referred to as data-control plane interface (D-CPI), defines instructions for the forwarding devices on how to process data packets. Candidates for such an interface are OpenFlow [48], NETCONF [49], or OVSDB [50]. OpenFlow, initiated by Open Network Foundation [51], is the first standardized programming interface and becomes the most popular protocol in SDN [19]. OpenFlow defines a flow as a sequence of packets that share the same *match*, that is, some configurable set of characteristics. Match field combination enables different manners of flow matching, e.g., on source-destination MAC/IP pair and on a certain ingress port. With the OpenFlow protocol, the SDN controller can reactively and proactively add, update or delete flow entries in the flow tables [52].

OpenFlow Switch. According to the latest OpenFlow version 1.5.1 [52], a typical OpenFlow-enabled switch is composed of one or more flow tables and a group table, and one or more OpenFlow channels to external controller(s). Each flow entry in the flow table consists of match fields, counters and a set of instructions for successful matches. The flow tables are recommended to be implemented as Ternary Content Addressable Memory (TCAM) [48], [53], which supports flexible wildcard matching. When a data packet enters the switch, header matching starts at the first flow table and might proceed to other flow tables in the pipeline. The first matched entry then defines how should the packet be handled, such as forwarding to another port, flooding to all ports, dropping directly and forwarding to the controller. If no match is found in a flow table (i.e. table-miss), the packet will be processed depending on the default configuration, e.g., send to controller or discard.

In order to facilitate the evolution of OpenFlow and other SDN switches, programmable switching platforms have been proposed. Namely, P4 [54] (Programming Protocol-independent Packet Processors) was proposed to enable the programmability of data plane behaviors. Independent from protocols, programmers could customize the way that packets will be processed and further optimize the performance of network service.

Controllers. There are several SDN controller implementations that run on different platforms and feature various characteristics. For example, NOX [55], Ryu [56] and Floodlight [57] explore the possibility of different programming languages (C++, Python and Java) and of single/multi-thread processing. To overcome single-point-of-failure, performance and scalability concerns [58], [59], physically distributed controller implementations have drawn much more attention recently. Onix [60], as one of the first proposals, distributes the network state over multiple instances. On the other hand, controller instance in HyperFlow [61] stores the whole network state and therefore different instances can be treated as backups of each other. ONOS [62] and OpenDaylight [63], as two production-level SDN controller implementations that are supported by the Linux Foundation [64], natively distribute the control logic into several instances. Network operators are allowed to choose what information to synchronize and which



(a) SDN: The logical centralized controller cluster controls the underlying forwarding devices. The southbound API (i.e., D-CPI) defines the interface between control and data plane. By calling the northbound API, the applications push dedicated flow rules to the devices to realize different functionalities.

(b) NFV: The infrastructure manager virtualizes the physical infrastructure, i.e., commodity servers, and exposes abstracted CPU, memory and I/O resources to the orchestrator. When receiving a function request from the VNF manager, the orchestrator instantiates and configures the function accordingly.

(c) NV: The physical substrate network is abstracted as link and node resources. Under resource constraints, the VN topologies are embedded in the substrate towards certain system objective. Because of isolation, different VNs could run different protocols or controlled by different applications.

Fig. 2. Conceptual illustration of SDN, NFV and NV..

level of consistency [65] should be followed, according to their preferences over synchronization traffic overhead [66], convergence time [67], and behavior correctness [66]. Notably, when a controller instance fails, one of the other running instances will take over and maintain network operation.

Other Surveys. [4], [17]–[20] cover general aspects of SDN. Security of SDN, as an important issue, has been surveyed in several articles, such as [21]–[26]. The problem of control plane scalability has been surveyed in [68]. [27] provides a taxonomy that classifies the reviewed research works performed in SDN, whereas [69] presents a systematic survey of up-to-date OpenFlow-based SDN programming languages and [28] surveys network innovations with OpenFlow. Regarding merging SDN with wireless, the work of [29] and [30] surveys general aspects in wireless and the work of [25], [31]–[33] focuses on SDN mobile networks.

B. Network Function Virtualization

Concept. Traditionally, network operators deploy physical and proprietary equipment (also known as "middleboxes") and chain the equipment in the network, i.e., guide traffic through each of them with or without an order, as to achieve a full network service. With a rising diversity of requirements and demands of network services, operators often require many different network hardware devices, which potentially leads to high deployment and operational costs. In order to address these challenges, following the thrive of virtualization techniques, NFV enables softwarization of network functions, i.e., Virtual Network Function (VNF). VNFs can be deployed on commodity commercial off-the shelf (COTS) (e.g. x86 based) hardware [70]. This leads to flexibility in network function deployment and service innovation by reducing the effort and time to design, deploy and manage various network functions/services. It also allows network operators to flexibly assign resources to different VNFs in the field, according to the users' traffic and demands or according to the operator's objectives, e.g., consumed energy or load balancing.

NFV Use Cases. Several use cases have been proposed in [71] by ETSI. One of the main targets is the virtualization of mobile networks, including both Mobile Core Network (MCN) and Radio Access Network (RAN). The 3GPP has standardized the network functions in the LTE MCN, also referred to as Evolved Packet Core (EPC), which facilitates their implementation in software. Some examples of these core network functions are the Mobility Management Entity (MME), which manages the mobility of users, and the Serving and PDN-Gateways (S/P-GW), which forward user data traffic under the negotiated and predefined policies. These mobile core functions can be implemented as different VNFs and scaled up or down according to their own requirements. This can be done to cope with the expected increase in mobile user traffic, for instance. Additionally, most of the functions of the mobile RAN are also standardized and hence can be virtualized. In this case, virtualization can achieve higher resource efficiency and flexibility for the radio resources. Namely, NFV enables the pooling and consolidation of the baseband processing of the eNodeB, including PHY, MAC, RLC and PDCP layers.

There are several other promising use cases that target

function virtualization, such as Content Distribution Networks (CDN) [71]. The rise of video streaming poses great challenges on network operators, because of massive traffic volume and strict QoS requirements. When receiving a request from a user, the CDN controller selects a cache node, mostly in geographical proximity of the user, and then redirects the user to the VNF cache node. Since the user traffic demands follow a predictable pattern, hardware resources can be dedicated to other VNFs during weekday business hours for instance. It is also more flexible to configure new VNF cache nodes in face of new delivery requirements.

Other Surveys. The general aspects of NFV have been surveyed in [5], [34]. Similar to SDN, security issue is also a concern and is surveyed in [35] and [36]. Particular for the core network, the work of [37] surveys different virtualization architectures. The function placement problem is thoroughly surveyed in [38].

C. Network Virtualization

Concept. Network virtualization is the technology that provisions multiple logically isolated networks, which may own distinct addressing schemes and forwarding policies, in the same physical infrastructure, i.e., nodes and links. From the operators' point-of-view, NV splits the role of Internet Service Provider (ISP) into two parts: infrastructure provider and service provider [72]. The infrastructure provider manages the physical infrastructure and provides virtual networks for the service providers. The service providers in turn interconnect end-users and offer networking services. Indeed, NV is flexible in acquiring and changing virtual networks on-demand depending on the users' traffic and requirements. It also supports the infrastructure providers with the flexibility to lease their resources to different services providers, expediting cost savings of operation.

Implementation. There are several approaches and technologies which enable the virtualization of network links and/or nodes. Virtual Local Area Network (VLAN) for instance allows a set of local networks, whose broadcast domains are isolated, to coexist in the same physical network. The packets of each network are recognized by the additional VLAN headers indicating the VLAN ID. Moving further, overlay networks are logical networks that are directly deployed in physical networks, based on protocols like GRE and VXLAN. The above implementations focus merely on the virtualization of network links, whereas in a fully virtualized network, both links and nodes should be virtualized [72]. In other words, logical node resources, e.g., CPU processing and memory, are assigned to the different tenants. Node virtualization is normally realized by network hypervisor [73], which can be deployed locally on the nodes or remotely manages the resources of multiple network nodes.

Other Surveys. The general aspects of NV have been surveyed in [6]. Various VNE algorithms are discussed in [39]. Besides the wired networks, the VN concept has also been introduced to wireless area and thus provides much potentials, which are surveyed in [40], [40], [41]. A particular focus on 5G Mobile Network is unfolded in [42] and another focus on cloud computing is elaborated in [43]. The work of [44] surveys hypervisor implementations and proposes a performance evaluation architecture.

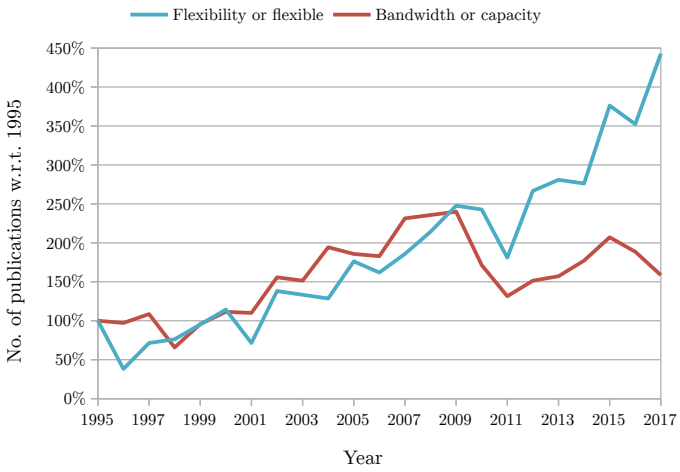


Fig. 3. Evolution of the number of publications containing the words "flexible" or "flexibility" in contrast with those containing "bandwidth" or "capacity" in four major IEEE journals and magazines on communication, with respect to the number of publications in 1995.

III. Towards a Classification for Flexibility

In recent years, flexibility has become a buzzword in communication networks and has drawn more and more attention of researchers. As an illustrative example, we perform a small study on the literature mentioning flexibility. We look for publications containing the words "flexible" and/or "flexibility" in their titles, abstracts or keywords in the *IEEE Communications Magazine*, *IEEE Transactions on Communications*, *IEEE Wireless Communications*, and *IEEE Transactions on Wireless Communications* from 1995 to the time of this survey. We compare the evolution of the popularity of flexibility with that of the words "bandwidth" and/or "capacity", which are related to the performance of a network from another perspective. Fig. 3 shows how these terms evolve over time, when compared to the number of publications in 1995. We can see that bandwidth-related words were increasingly used from around 2000 to 2009, but this trend has not continued in recent years. On the contrary, the use of flexibility-related words has been increasing consistently in the last decade, peaking in 2017.

This study illustrates the growing interest in flexibility for communication networks and even shows a change of interest towards flexibility. When looking deeper into the literature, we also notice the lack of consistency in defining the term "flexibility" in various state of the art proposals. With this survey, we intend to provide a common ground for understanding what flexibility really means and how to compare different design

choices of flexible networks.

A. Definition of Flexibility

The lack of a rigorous definition of flexibility has produced lots of different interpretations in literature. For instance, some researchers understand flexibility as the ability of network operators to change the network configuration with ease, whereas others consider that their design is flexible because it can dynamically adapt to different kinds of traffic. In any case, almost no prior work explicitly states its definition of flexibility. Our intention is to propose a definition of flexibility which is based on the most common understanding to use it throughout our survey.

Flexibility analysis is always specific to the particular system under study, which in our case is communication networks. Therefore, a prior understanding of the elements of a communication network is required. A communication network is composed of *topology*, *flows*, node *functions*, and *resources*. The topology can be represented as a set of nodes and links. Flows are defined as source and destination node pairs, together with the respective data rates. Node functions describe the specific actions applied to the network traffic, e.g., firewalls and load balancers. Resources, e.g., computation power, memory and bandwidth, are also mandatory for the operation of networks. Table I demonstrates that, due to conceptual distinctions, SDN, NFV and NV focus on different subsets of these elements and support flexibility there in particular. For instance, SDN enables the manipulation of the traffic forwarding paths in the network, whereas NV empowers multiple isolated networks with diverse topologies in the same physical substrate.

In order to incorporate as many notions from the literature as possible, we define network flexibility as the *timely support of changes in the network requirements*. For the sake of brevity, we refer to "changes in the requirements" as *requests*. We consider two different manners in which a network could support these requests. First, the network may be designed in a way that it simply accommodates the requests without major adaptations. Alternatively, the network may need to adapt the state of the topology, flows, functions, or resources to match the new network requirements. When the adaptation happens, it should meet a time constraint, i.e., the adaptation should be quick enough and finish in time.

Let us illustrate this definition with an example. Consider the emergence of new user demands in an enterprise network, which causes a traffic increase. We would deem the enterprise network design as flexible if this traffic increase can be accommodated within a short time frame. Using our terminology, the traffic increase is translated into a request. When having enough node and link capacity, the network can accept the new traffic without any adaptation. Otherwise, the network should possibly modify the state of flows, functions, or resources (e.g., rearrange flows) and thereafter support the traffic within a predefined time constraint. In both cases, the network will be flexible according to our definition.

The concept of *request* is important in the analysis of flexibility, since it triggers the demand for flexibility. Based on the above, we consider a *request* as a *change in the network requirements that may imply a modification of the topology, flows, functions, or resources*. Accordingly, there are many possible sources of requests: traffic variations, user mobility, network upgrade, etc. A comprehensive list of all the possible request origins will help us to perform an extensive analysis of network flexibility.

B. Flexibility Categories and Aspects

Although these elements are conceptually very different, the ways of changing them are yet similar. Indeed, there are three basic abstract operations that are applicable to them.

Firstly, we can *adapt the configuration* of the network to accommodate requests. The configuration can be a single parameter setting, an installation of network flows, or an outline of how network functions behave. In turn, the adaptation could be a change of parameters, i.e. "re-configuration", or an addition to the possible set of supported configurations.

Second, thanks to virtualization techniques, functions can be dynamically placed in the network to support both latency and service requirements. Although the placement may affect how functions perform in some cases, in principle they can be moved across the topology while keeping their configuration intact. Therefore, this operation is to *locate functions*.

Third, the most general operation is to *scale* network elements, which means adding or removing functions, links, nodes, or resources. The ability to scale its elements enables the network to increase or decrease link capacity, reserved bandwidth of particular flows, allocated computation power of network functions, as well as the number of network function instances.

These three operations can be used to classify systems with respect to flexibility. Indeed, we observe that a network can be assessed in terms of its flexibility *(i) to adapt the configuration of its flows and functions*, *(ii) to locate functions*, and *(iii) to scale its resources, functions, or topology*. As a result, we henceforth refer to these operations as *flexibility categories*, which are shown in Table II.

Flexibility categories provide initial valuable insights on the similarity of proposals that are built upon different technologies. Nonetheless, if we want to study the kind of flexibility that is related to a specific technology, we need to look finer granular. For a finer segmentation, we define the concept of *flexibility aspects*. Flexibility aspects are the applications of flexibility categories to topology, flows, functions, or resources of the network. In this way, adapt configuration is split into *flow configuration* and *function configuration*; locate functions is reformulated as *function placement*; and scale is split into *function scaling*, *resource scaling* and *topology adaptation*. Additionally, we define *parameter configuration* as distinct from *function configuration*, reflecting the ability to change the operation parameters of a function instead of the whole functionality.

TABLE I
COMPARISON OF SDN, NFV AND NV TECHNICAL FOCUS AREAS W.R.T. FLEXIBILITY SUPPORT.

	Flows	**Functions**	**Resources**	**Topology**
SDN	Routing, traffic engineering, flow monitoring	–	Flow bandwidth, control plane adaptation	–
NFV	–	Dynamic provisioning and placement, service function chaining, monitoring	Adaptation of computation, memory, I/O	–
NV	–	Virtual network implementation	Virtual node capacity and link bandwidth embedding	Virtual network topology adaptation, multi-tenancy, isolation

TABLE II
TECHNICAL CONCEPTS AND THEIR SUPPORT OF FLEXIBILITY IN NETWORKS. (✓: MAIN TARGET)

Category	**Aspect** (see Sec. III-B)	**SDN**	**NFV**	**NV**
Adapt configuration	Flow Configuration: flow steering	✓	-	-
	Function Configuration: function programming	-	✓	-
	Parameter Configuration: change function parameters	-	✓	✓
Locate functions	Function Placement: distribution, placement, chaining	-	✓	✓
Scale	Resource and Function Scaling: processing and storage capacity, number of functions	✓	✓	✓
	Topology Adaptation: (virtual) network adaptation	-	-	✓

The definition of each flexibility aspect becomes clear when we look into the technologies of softwarized networks. For instance, for configuration SDN enables programmability of flow forwarding, aided by a global view of network topology and traffic status, thus provides flexibility in *flow configuration*. NFV mainly provides flexibility in *function configuration*, as it virtualizes network functions and makes them deployable on commodity hardware. Besides, the parameters of such functions can be easily tuned, thanks to software implementation, leading to flexibility in *parameter configuration*. The virtualized networks in NV, especially mobile networks, typically possess a number of parameters, which can be adjusted with a target of higher transmission throughput. That being said, NV enables flexibility mainly in *parameter configuration*.

The location of network functions plays a vital role in a system's performance. Obviously, softwarized functions can be more easily placed on different servers, compared with hardware middleboxes, therefore enables the flexibility in *function placement* in NFV. Virtual Network Embedding (VNE) handles the procedure of allocating virtual nodes, which we refer to as "functions", and it guarantees the flexibility in *function placement* in NV.

Scalability issues are also tackled by network softwarization technologies. SDN, NFV and NV all provide flexibility in *resource* and *function scaling*, however with different focus. In SDN, a resource typically means the bandwidth assigned to a particular flow and functions are in the form of controllers. Virtualized functions in NFV have both properties of assigned computation resource and deployed instance amount. Embedding virtualized networks onto a substrate employs two strategies. The first one is to adapt the allocated resource, i.e., bandwidth, for virtual links. The other one is to change the mapping itself, migrating virtual links to other physical links, which poses the flexibility in *topology adaptation*.

As a final step in the confection of our list of flexibility aspects, we merge together *resource* and *function scaling* based on their similarity. Although conceptually different, in practical applications there are not many differences between providing a VNF with more resources and creating a new VNF that uses the additional resources. Based on our explanations, we define six flexibility aspects as illustrated in Fig. 4 :

- **Flow Configuration:** Creation, removal or adaptation of the course of flows. The steering of the flows inside a network is usually performed by configuring forwarding policy for a flow on each network hop. An example of flow configuration is depicted in Fig. 4a.
- **Function Configuration:** Adaptation of the functionality of network elements such as firewalls, NATs, proxies, or load balancers, for example through software virtual network functions or programmable bare metal switches. Fig. 4b shows an example of function configuration.
- **Parameter Configuration:** Adaptation of the values and policies applied by each network function. This means that network functionality remain the same, however, the parameters configured on those functions can vary. An example of parameter configuration is depicted in Fig. 4c.
- **Function Placement:** Adaptation of the location of network functions. The function placement has a direct impact on the network performance, e.g., the SDN controller placement with respect to switches affects control latency. A representation of function placement is shown in Fig. 4d.
- **Resource and Function Scaling:** Adaptation of the

assignment of network resources to flows or functions, or adaptation of the number of deployed instances of a specific function. For instance, extending the assigned processing and storage capacities to a load-balancing function, or increasing the number of SDN controllers. Fig. 4e shows a simple example of resource scaling.
- **Topology Adaptation:** Adaptation of the graph structure of the network through adding or removing nodes or links. An example of topology adaptation is depicted in Fig. 4f.

Table II gives an overview of the categories and related aspects, as well as the major flexibility aspects that each technology supports. The concept of flexibility aspect is very useful to dissect the flexibility of softwarized network designs. Owing to this, the six aspects will be used as criteria to classify the publications that we incorporate in this survey.

C. Cost of Flexibility

The price to pay for flexibility in networks is related to the resources and guarantees that are needed to realize and operate a more flexible system design. For example, more data centers have to be installed to support function or controller migration. In addition to infrastructure cost including its operation, the costs emerging from the reaction to changes, such as migration overheads, need to be considered.

At the moment, we do not know how exactly costs are related to flexibility and we expect that this will also vary with designs and flexibility aspects. We rather argue at this state of research that it is not enough to come up with new network designs for flexibility, but the flexibility vs. cost trade-off has to be considered as well. Intuitively, one might think that costs rise with increased flexibility. However, a more flexible design can lead to cost reductions in the longer term. For a quantitative analysis and, in particular, for the comparison of different design choices, we need to consider all cost factors that are related to flexibility.

According to our previous work [1], we distinguish four main categories of cost in relation with flexibility. To provide more flexibility, additional resources, e.g., data centers, might be required (CAPEX) and the respective systems might employ a more complex mechanism to support adaptation, which also translates into cost. Operational cost (OPEX) comprises all costs for the operation of a network that might increase due to flexibility, e.g., control or data plane latency. In addition to the above normal operation cost, we have cost involved in the adaptation process, e.g., synchronization overhead or configuration latency. Moreover, costs might occur due to the violations of SLAs, when adaptation takes longer than expected.

Owing to the high importance of the cost, it should always be considered in every flexible network design. In our analysis of the state of the art, we pay particular attention to whether the cost of the adaptation is addressed by the proposals.

D. Flexibility in Non-softwarized Communication Networks

At the beginning of this section, we demonstrate that the term "flexibility" is growing in popularity within the field of communication networks. This is in line with the increasing popularity of softwarized networks, since both concepts are highly correlated. However, this does not mean that flexibility is unique to softwarized networks. Indeed, a number of techniques used by non-softwarized communication networks are used to increase their flexibility.

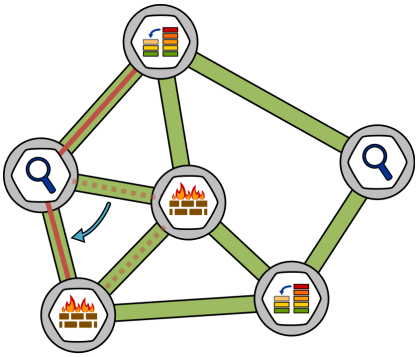
In legacy mobile networks, examples of flexible networks are *self-organizing* or *self-optimizing networks* (SON), which have been implemented in LTE networks [74]. *Self-organization* refers to the ability of the network to configure itself in order to properly operate, whereas *self-optimization* takes the configuration one step further so that the operation is enhanced [75].

A self-organizing LTE network is set to allow for automatic configuration of new base stations, so that the operator can extend the network easily, thus increasing the flexibility of the network. After the connection, the base station is able to acquire the software to operate along with the required parameters, such as the cell identity, power settings, tracking area code, etc. For that, the base stations are prepared to configure themselves according to the data they receive from the core network and from the transmissions they sniff from neighbor cells.

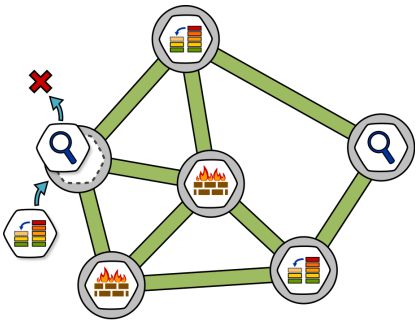
Once the initial configuration is done, base stations are also able to optimize their operation in an autonomous fashion. For instance, they can dynamically agree with other neighbor cells on the configuration of the random access channel. This channel is used for initial uplink access and it is particularly prone to suffer from interference. A more advanced technique is *inter-cell interference coordination*, through which the base stations can dynamically adapt their scheduling to the load of neighbor cells. In order to do so, the base stations can be configured to exchange information about the transmitted power and received interference in each resource block.

Self-optimizing networks also cover load balancing between base stations, which may increase the total utilization of the network and its ability to adapt to traffic changes. For this, the base stations can ask others about their load and perform a handover to a less loaded cell if needed. A similar technique can be used to improve the network configuration after a link failure. In case of a radio link failure because of an early or late handover, the mobile will reconnect to the old or a new cell, respectively. In either case, both cells exchange information to improve their handover thresholds and avoid the same problem in the future. This technique improves the robustness of the network. Overall, all the mentioned techniques enhance the ability of the network to support changes, thus increasing its flexibility.

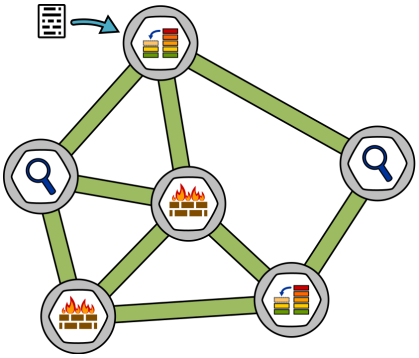
Besides wireless networks, wire line networks also target flexibility to some extent. Routing algorithms can be flexible in adapting to different topologies and traffic conditions and still achieve satisfactory performance [76]–[78]. In multicast,



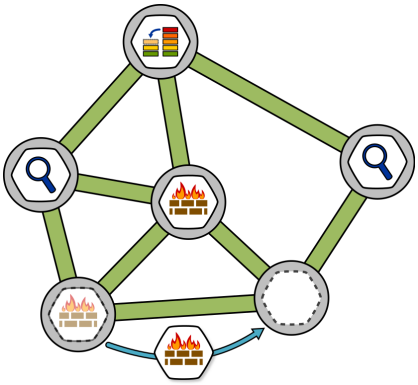
(a) Example of flow configuration. The old flow path (dashed line) is replaced with a new one (solid line). The endpoints remain the same, but the intermediate node have changed.



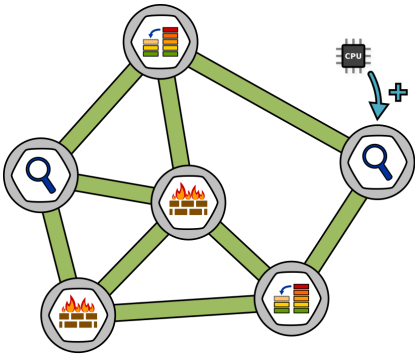
(b) Example of function configuration. A DPI function is being replaced by a load balancing function in the left most node.



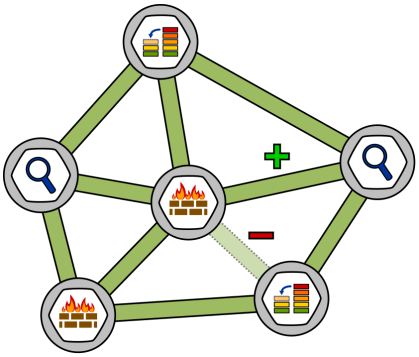
(c) Example of parameter configuration. A new policy is being loaded into the load balancing function of the topmost node.



(d) Example of function placement. The firewall function in the lower left node is being moved to lower right node.



(e) Example of resource scaling. Additional CPU resources are being allocated to the DPI function in the right most node.



(f) Example of topology adaptation. The link connecting the central and the lower right node is being deleted, and a new link connecting the central and the right most nodes has been created.

Fig. 4. Depiction of flexibility aspects in a simple network. Grey circles represent nodes, thick green lines are links, narrow red lines represent flows (when relevant), and white hexagons represent functions within nodes. Three different functions are considered: firewall, Deep Packet Inspection (DPI), and load balancer.

flexible allocation of forwarding states among routers [79] can potentially balance the utilization of routing tables. Routing in overlay multicast tree is also more flexible than legacy IP multicast tree [80]. Moreover, networking protocols are also enhanced with the consideration of flexibility, and the examples are P2P [81], [82], Neighbor Discovery [83], BGP [84] and OSPF [85].

There are many other examples of flexibility in legacy networks, since adaptation to changes has been always a desirable characteristic of any network. Nonetheless, in this survey we limit the scope to softwarized networks, as these are specially related to flexibility and already have a large state of the art for a solid analysis.

E. Flexibility in Other Areas

Flexibility is a concept that is present in many fields besides communication networks. A survey on flexibility, even if it is focused on softwarized networks, would not be complete without an outlook on other fields. Therefore, we present a brief overview on the flexibility in a variety of research fields.

1) Economics: In economics, decision-making is a very important research field, in which flexibility is a very common term [9], [10]. Any decision or plan implies risk because it deals with uncertain factors. The ability to adapt to changes in these uncertain factors is very important in order to avoid losses. For instance, in [10] the authors address the definition and measurement of flexibility of financial decisions. They measure the final result of such decisions with some *attribute*, such as money. Then, they model the beliefs about the attribute of the uncertain results with what they call a *prospect*. Finally, they define the *certain equivalence* as the money that the decision-maker would take instead of making the risky decision. When comparing two different prospects, the one with the higher certain equivalence is more flexible. Although this definition and measurement of flexibility is rather thorough and suitable for financial decisions, it is hardly possible to translate it into communication networks.

2) Manufacturing: Flexibility in manufacturing has been a major concern since the 1950s, as manufacturers want to adapt efficiently to changes in the demand, raw materials,

regulations, etc. There are several publications that survey the vast literature in this topic, such as [86] and [87]. In [87], the general concept of flexibility of a system is defined as "its adaptability to a wide range of possible environments that it may encounter". Moreover, they define flexibility in manufacturing as "being able to reconfigure manufacturing resources so as to produce efficiently different products of acceptable quality". They define 11 types of flexibility: machine, material handling, operation, process, product, routing, volume, expansion, program, production, and market flexibilities. For each type, they propose a definition and a measure. These measures are often ratios between some feasible quantity and the ideal one. Measures of manufacturing flexibility are also the exclusive focus of some other publications. Some examples are [88], which is dedicated to their formal definition, [89], which proposes a framework to facilitate their development, and [90], which addresses the problem of measuring supply chain flexibility with updated methods.

3) Management science: Management science is also one of the fields in which flexibility has been widely studied, since the ability to anticipate and adapt to changes is one of its main objectives. However, they suffer from the same vague definitions and measurements of flexibility. Abundant research has been done to define, categorize and measure it [11]–[13]. One example is [13], in which the authors propose a comprehensive definition and metrics for flexibility. They identify four dimensions of flexibility: *time*, *range*, *intention*, and *focus*. According to the *time* it takes for a system to respond to changes, they divide flexibility into *operational* (short term), *tactical* (medium term), and *strategic* (long-term) flexibility. An alternative division is based on the *range*, that is, on the number of options that a system have in order to adapt to a change: *foreseen* flexibility (for likely changes) and *unforeseen* flexibility (for unlikely changes). Furthermore, there are two types of flexibility regarding *intention*: *offensive* and *defensive* flexibility. Finally, they identify the *focus*, i.e., the area in which the flexibility is created, as the fourth dimension, which motivates yet another division: *internal* and *external* flexibility. In order to measure flexibility, the authors propose a set of four metrics: *efficiency*, *responsiveness*, *versatility*, and *robustness*.

4) Software engineering: Software flexibility is addressed in [14]–[16], and a software can generally be tagged as flexible or inflexible. In [16], flexibility is treated in a similar manner to complexity. Indeed, the authors aim to model the *evolution complexity* of the software. In order to define evolution complexity, they base on an *evolution function*, which maps the old problem, old implementation and shifted problem into the adjusted implementation. Then, they define a *cost function* relating the evolution function to the effort needed to make and adjustment. Depending on whether the complexity of the cost function is constant or linear, they call the software flexible or inflexible, respectively. The flexibility of software is closely related to that of communication networks. In fact, the emergence of softwarized networks is the main cause of the interest in flexibility within communication networks.

After reviewing all these different fields, we observe that flexibility is a desired property of a system, and it is often regarded as an improvement over previous designs. Moreover, we notice that our notion of network flexibility is also consistent with other notions that exist in other fields. Finally, we observe that the main problem in other fields is the lack of a rigorous definition and a homogeneous quantitative measure. In the following analysis, we see that this is also the main challenge with softwarized networks.

IV. Analysis of Flexibility in the State of the Art

In this section, we present a comprehensive analysis of flexibility in the literature based on a selection of publications regarding softwarized networks. Our list of surveyed publications includes publications that explicitly claim flexibility, either mentioning flexibility as one of the design goals or as a consequence of the design, and publications that meet our definition of flexibility without using this term explicitly. In this way, we are able to investigate flexibility in the literature based on content and not on terminology.

Our scope covers all kinds of softwarized networks, including wireline networks, data center networks, and mobile networks (mobile core network and radio access network). Furthermore, the publications leverage flexibility for various purposes, such as policy enforcement, load balancing, failure recovery, throughput enhancement, deployment cost reduction, etc. To present the publications, we follow a structure that combines the three categories of flexibility and further the aspects that reside in different categories (defined in Sec. III-B) with the different network softwarization technologies (introduced in Sec. II). A summary of this classification is shown in Table III. A more comprehensive table showing all surveyed papers with their main objectives, as well as adaptation costs and time, is shown in Table VII at the end of this survey.

A. Adapt Configuration

1) Flow Configuration in SDN: SDN, in principle, furnishes opportunities to perform adaptive routing, load balancing and link failure recovery. This is accomplished by creating, deleting and re-configuring flows within the forwarding devices.

Nguyen et al. [91] define an optimization problem that maximizes the total amount of traffic delivered to the destination nodes, under the constraint of restricted TCAM in an SDN switch. Due to flow configuration flexibility of SDN for changing traffic situation, traffic can be separated into flows for which policies can be implemented via flow rules and others for which policies cannot be satisfied and which have to be routed to the controller. The authors argue that the flexibility of optimizing such flow configuration comes at an intensive computation cost, as the optimization problem is NP-hard. Besides, extra forwarding latency is expected due to controller invocation of unsatisfied flows, which leads to additional cost.

The work in [92] considers traffic engineering in a hybrid network running SDN and legacy OSPF, and leverages flexible traffic flow distribution for changing traffic loads. An

TABLE III
CLASSIFICATION ACCORDING TO FLEXIBILITY CATEGORIES AND ASPECTS

Category	Aspect	Publications
Adapt Configuration	Flow config in SDN	[91]–[109]
	Function config in NFV	[103], [104], [108], [110]–[113]
	Parameter config in NFV	[112], [114]–[117]
	Parameter config in NV	[118]–[121]
Locate Functions	Function placement in NFV	[99], [107], [109], [112], [122]–[135]
	Function placement in NV	[136]–[140]
Scale	Resource and function scaling in SDN	[141]–[144]
	Resource and function scaling in NFV	[113]–[117], [145]–[152]
	Resource and function scaling in NV	[106], [153]–[160]
	Topology adaptation in NV	[138], [161]–[165]

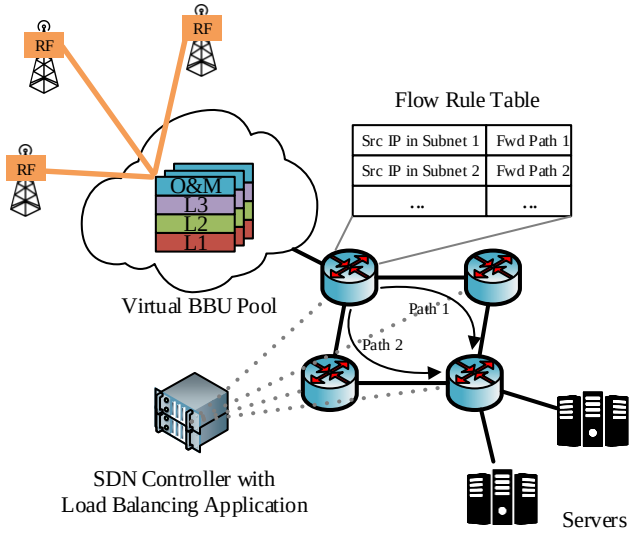


Fig. 5. Example of configuration adaptation. The left hand side demonstrates Cloud-RAN [113], in which the Baseband Units (BBUs) are centralized as a pool so as to achieve cost savings on baseband resources. The remote radio heads are connected to the BBU Pool via fiber, and the RAN parameters can be dynamically configured. The right hand side is an SDN network, whose forwarding behavior is defined by the load balancing application in the controller. Based on the current traffic status, the controller reconfigures the flow rules in the data plane.

optimization problem is defined to minimize the maximum link utilization. Whereas flow forwarding in OSPF-enabled nodes follows a strict shortest-path strategy and flows entering an SDN-enabled node can flexibly be allocated to different possible outgoing links, so that the global traffic distribution becomes more balanced.

Lee et al. study the routing problem in traffic engineering and propose a resource preference aware routing scheme [93]. The flexibility of configuring flows with SDN helps to balance the load in the network, which is illustrated conceptually on the right-hand side in Fig. 5. Two types of resource are identified while making routing decisions, i.e., link bandwidth usage and flow table usage. First the analytic hierarchy process analyzes the characteristics of network traffic and quantifies the resource preference. Second, the k-shortest path algorithm creates a list of candidate paths. With the resource preference value, the RPA-RA algorithm evaluates the path candidates and returns the best candidate.

For network management, SDN can also enforce dynamic traffic policies in a flexible manner, under changing network statistics. [94] proposes a policy-based management approach which gathers the monitoring statistics and reconfigures the data plane forwarding accordingly. Monitoring statistics are the context to tune predefined policy templates, and based on the templates, the controller generates and pushes unique flow rules that accommodate the current traffic situation. The authors evaluate a link failure scenario, and emphasize the time factor in adaptation: traffic redirecting to the backup server should happen within a short time. However, they scale time only in “periods”, ignoring the actual values.

To help flexibly route traffic and mitigate frequent controller invocation of new flows, Su et al. leverage machine learning techniques and blocking island paradigm and propose the CheetahFlow scheme [95]. A support vector machine predicts frequent communication pairs and the controller sets up wildcard flow rules accordingly. Blocking island represents the network topology in the form of a tree and demonstrates available bandwidth information. Experiments show that the framework can significantly reduce forwarding latency and decrease the number of flow rule entries in the switches.

The work [96] studies routing in data center networks. Applying flow configuration flexibility to a multi-path routing scenario, the authors focus on decreasing the involved cost of route adaptation. Whenever the controller detects over-utilization of a certain link, it automatically calculates a list of candidate routes, which bypass the congested link, and choose the one with lowest congestion level. The overhead of managing flow rules is considered as a cost factor for adaptation, and such cost grows significantly when flow migration is frequent. To solve this issue, the Segmented Wildcard Forwarding scheme potentially shrinks the number of flows

that need to be updated and can still achieve a high overall throughput.

Cascone et al. advocate enforcing some of the forwarding policies directly at the data plane [97], i.e. SDN switches, with the support of OpenState [166] that allows packet processing in a stateful manner. In the application that guarantees forwarding consistency, OpenState allows a flexible choice of granularity and lifetime of a forwarding decision. Experiments indicate a significantly smaller switch processing latency than in the case of invoking the controller every time a flow emerges.

By decreasing the convergence time, SDN enables faster recovery from failures. [98] shows that rule installation time during flow setup is significant and thus suggests a new systematic architecture MAZU, which "provides operators additional flexibility in designing schemes to better meet failover requirements in their networks". The packet-ins of new flows are redirected to a fast proxy that takes over the task of interacting with the controller. Furthermore, an algorithm running inside the controller orders the flow rules before pushing them to the switches in order to further shorten the installation time. The latency to update the network state (adaptation) upon failure refers to a time constraint with the target adaptation time in the order of tens of seconds. Implementation based evaluation shows that MAZU can decrease such latency by up to 5 times.

To enable a flexible network with middleboxes of abundant functionalities, several works propose the integration of SDN and NFV. One example is [99], which simultaneously optimizes the maximum link usage/CPU utilization and the maximum delay of flows. A joint optimization model places network functions and embeds flow paths at the same time. Here the flexibility in flow configuration serves as a supporting aspect, as after placing network functions, routing is mandatory to guide data traffic to traverse those functions.

Another work in this direction is SDNFV [100], in which the authors claim that SDN alone cannot tackle complex packet processing functionality, such as proxies and deep packet inspection, which rely on a per-packet process and have to be performed by virtual network functions. To demonstrate the efficiency and flexibility of SDNFV, a prototype monitors flow statistics and once it detects elephant flows, it re-orchestrates the service chain and re-routes other mice flows to decrease the average latency.

Other than data plane flows, control plane flows can also be potentially configured in a dynamic manner. Basta et al. consider virtualization of SDN networks, where distributed hypervisors support the isolation of tenant networks [101]. As the network requirement changes, the adaptation of hypervisors' locations and the reroute of control paths enable a more flexible control plane. The authors clearly state the adaptation time as the time to reroute control paths, and they design an elaborate protocol to speed up path migration. Prototype evaluation shows that the migration takes maximum 50 milliseconds, and during the migration execution, the average control latency increases.

[102] considers a combination of SDN and IoT (Internet of Things), where flexible flow management for the control plane is as important as that for the data plane. The authors propose a game theory approach to increase the controller's utility and reduce latency simultaneously. They calculate a pay-off function of each controller instance and forward it to a global decision maker which decides when and where to offload control traffic.

Flow configuration also plays a role in several mobile network scenarios. Mobile networks usually experience large variability in user distribution and traffic demand, which demands for flexibility as a somehow mandatory characteristic. This explains why a large amount of research effort has been put into designing and implementing Software Defined Mobile Networks (SDMN). Nonetheless, proposals featuring flow configuration in SDMN are less abundant than in non-mobile SDN domains. The reason is that flows in the wireless network have much less alternative paths, i.e. single path between base station and mobile device, than in the wireline mobile core network, for example. Nevertheless, there are notable Software Defined Mobile Networks (SDMN) that have been proposed featuring flow configuration focusing on the core part.

The first example is CellSDN [103], which is an SDN-based architecture for the LTE core network. Four extensions help to adopt the SDN architecture in the mobile scenario: routing policies based on subscriber attributes instead of IPs, presence of local agents at the switches to enhance scalability, ability to configure actions at the switches, and network virtualization. The first extension groups devices according to their attributes, such as cellular network provider or device type. After grouping, the network can deal with large flows of similar characteristics, instead of small, single-device flows. Then, the controller can decide how to route those flows in order to fulfill the QoS requirements.

In [104], the SDN-based mobile core network architecture SoftCell is presented. This architecture provides flexibility to the mobile core network to be prepared for changing traffic and network increase via reconfiguration of data and control flows. Instead of the classical LTE core, in which the flow-forwarding functions (S-GW and P-GW) are performed on dedicated hardware, SoftCell proposes to use simple commodity switches connected to a central controller, which takes over routing decisions. The controller steers the flows through different switches and middleboxes (devices implementing network functions), according to their QoS requirements.

MobileFlow [105] is a further proposal to combine SDN with mobile networks. Here, SDN provides the ability to reconfigure the flows, thus granting flexibility to ossified, conventional mobile networks. The main feature of MobileFlow is the use of MobileFlow Forward Engines (MFFE), similar to OpenFlow switches, and MobileFlow Controllers (MFCs), similar to OpenFlow controllers, instead of the conventional architecture of the LTE core. MFFEs are able to route the data flows to different network functions (like DPI), PDNs, and the Internet, according to the decisions of the MFCs.

Akyildiz et al. present SoftAir [106], a comprehensive

description of a virtualized SDN architecture for both core and radio access networks. Among other characteristics, it includes the ability of flexibly steering the traffic across data-plane nodes to balance the load, support mobility and guarantee QoS. Within the core network, SoftAir proposes to use simple software defined switches and a network controller implementing re-routing algorithms to dynamically adapt the forwarding rules. For the RAN, the support of OpenFlow is envisioned in order to enable smooth transitions among different radio technologies, which will increase the number of options to re-route flows.

KLEIN [107] is another detailed SDN design of a flexible mobile core, which also features virtualization of functions. Its flexibility relies on a special resource management module, which can steer the traffic coming from users to the optimal data center. The goal is to balance the load among the data centers while meeting the delay, bandwidth and service constraints. To achieve this goal efficiently, the authors propose to aggregate user flows into group flows, hierarchically divide the problem into more convenient subproblems, and solve separately these subproblems for the control and data planes. According to the evaluation results, KLEIN is scalable and achieves nearly optimal flow management.

SoftNet [108] is a mobile network architecture that proposes SDN- and NFV-based core and radio access networks. It features flow configuration among multiple Radio Access Technologies (RAT), which makes the network more flexible when facing user mobility. Namely, it includes unified support for multiple RATs, enabling multiple alternative paths toward a single user. The authors of SoftNet emphasize that these paths can be used to set different flows, in order to offload traffic from the core network. This flow steering is performed by leveraging the SDN core network, whose control is centralized.

Finally, in [109] the authors present the 5G NORMA project, which provides a design for a highly flexible mobile network architecture following the SDN and NFV paradigms. Among other characteristics, it features flexible routing of flows to deal with traffic variations or application requirements. The routing task is mainly performed by a so-called Software-Defined Mobile Network Controller (SDM-C), which is one of the elements of the proposed control plane. Furthermore, re-routing motivated by user mobility is also assisted by the Mobility Management entity.

In summary, we observe that the advantages of flexible flow configuration in SDN have been applied to all kinds of SDN networks, including wired and wireless networks. The implementation of this aspect is the same among all of them, since it is a standard feature in SDN networks. Nonetheless, the goals of exploiting this aspect are different: maximize traffic delivered ([91], [167]), minimize link utilization ([92], [99]), balance the load ([93], [106]–[108]), dynamically adapt network policies ([94], [101], [104]–[106]), reduce switching latency ([95], [97], [99], [102]), decrease the adaptation cost ([96]), fast recovery from failures ([98]), or support numerous or complex virtual network functions ([100], [101], [103], [109]).

2) Function Configuration in NFV: Compared with hardware middleboxes, network operators would encounter less constraints in managing virtualized NFs that run in commodity servers. Indeed, virtualization techniques enable the flexibility to configure various functionalities at runtime.

Hwang et al. present a platform called NetVM [110] that allows middleboxes such as firewalls, proxies, and routers running in virtual machines, as a complement to the control plane capabilities in SDN. The flexibility of configuring various functions could fulfill the dynamic requests from the tenants. NetVM is supported by DPDK, which guarantees packet processing at near line speed. In order to facilitate different functions, abstractions of different network services are also imported to DPDK.

As with SDN, mobile networks have also benefited from the advantages of NFV. Virtualizing mobile functions allows them to be placed in a centralized data center instead of remote dedicated hardware, thus simplifying the management of core functions and paving the way towards enhanced inter-cell coordination. Since this characteristic perfectly complements the advantages of SDN, many SDMN proposals also feature NFV. This is especially prominent for mobile core network architectures, as the ones presented in the previous section.

The authors of CellSDN [103] consider an early form of function configuration in their SDN design. They do not explicitly mention NFV, since this concept was coined at the same time as the publication of CellSDN, but the underlying idea is that of NFV. Indeed, CellSDN describes how functions should be abstracted from hardware and flexibly located at the optimal positions. They argue that some (virtualized) functions, such as DPI or header compression, would be better performed at the switches than at the central controller. For instance, with DPI at the switches, it would be easier to identify applications and hence improve flow creation and routing. With that in mind, the authors consider that not every switch in the network should support these virtual functions, but only those configured by the central controller to do so.

SoftCell [104], which is introduced in the previous section, also features function configuration. Specifically, the SoftCell architecture allows to flexibly implement mobile network functions on off-the-shelf middleboxes. The authors suggest replacing all functions in the MCN with software implemented on commodity servers, which are referred to as middleboxes. In SoftCell, every middlebox can be configured to implement different functions, such as firewalls or video transcoders.

Following the lead of CellSDN and SoftCell, SoftNet [108] also includes the flexible configuration of virtual mobile network functions as one of its main features. Indeed, the authors describe functions dealing with communication control and network management, which can be turned on or off according to the state of the network. This ability is included in order to deal with different communication scenarios.

The use of NFV over mobile networks also extends to the RAN. In this regard, the most popular NFV-based RAN architecture is Cloud-RAN [113] (illustrated on the left-hand side in Fig. 5), upon which many other mobile NFV proposals are

built. This includes proposals featuring function configuration in softwarized mobile networks.

In [111], Sundaresan et al. present an architecture based on Cloud-RAN, in which Baseband Units (BBUs) can flexibly reconfigure their functionality, in contrast with the static configurations of Cloud-RAN. Specifically, BBUs can decide to use Fractional Frequency Reuse (FFR), Distributed Antenna Systems (DAS), or a combination of them. They claim that FFR is best suited for static users demanding high throughput, whereas DAS saves spectrum and power usage in the case of low-traffic mobile users. The selection is made in order to both fulfill the user requirements and minimize the resource usage. The time needed to perform the adaptation is also considered, since the RAN imposes strict delay constraints.

Another example of function configuration in mobile networks can be found in FlexRAN [112], a platform for implementing a software defined RAN. In FlexRAN, the control functions are softwarized and either delegated to a controller or handled by the eNodeBs. The location and operation of these control functions can be flexibly modified in order to face traffic changes or to meet operator decisions. Centralizing the control functions in the controller allows for a general view of the network, which provides better scheduling decisions. In contrast, distributing these functions over the eNodeBs enables faster responses for time-critical decisions. Furthermore, the authors present a highly configurable architecture, in which different implementations of the control functions can be dynamically updated in the network elements. This implies that the functions that are executed in the controller or in the eNodeB can vary according to the requirements of the operator and the state of the network. Although the latency between controller and eNodeBs is mentioned as a possible requirement, the latency caused by the function self-configuration is not directly considered.

To sum up, we see that function configuration in NFV is a lot leveraged in the area of mobile networks, although the concept is also investigated in other networks [110]. Within mobile networks, this aspect is exploited in both the core network to enable management and application functions complementing the core functions ([103], [104], [108]), and by radio access networks to switch between radio technologies [111], or dynamically vary RAN functions [112].

3) Parameter Configuration in NFV: Although NFV is first appeared with wireline networks, little prior work intentionally leverages or enhances flexible parameter configuration. Nonetheless, proposals of NFV mobile networks tackling parameter configuration are abundant. Many proposed Cloud-RAN-based networks dynamically seek the best configuration of their function parameters, especially when they are combined with other technologies such as Coordinated MultiPoint (CoMP).

CoMP is a technology that aims at wisely combining the transmissions of multiple base stations to improve the throughput and latency of data flows. Behind this simple underlying idea, a high number of parameters need to be carefully chosen: the set of base stations covering a user, beamforming parameters for each antenna of those base stations, transmission power, etc. Since an NFV architecture such as Cloud-RAN facilitates communication between base stations, configuring CoMP parameters in a dynamic and coordinated manner has become interesting for many researchers.

The first example of this interest is [114], where Ha et al. develop two algorithms for implementing CoMP over Cloud-RAN. They enable the configuration of the parameters of the functions handling CoMP in a flexible way, thus allowing the network to adapt to changes in the distribution of users, link qualities, etc. Specifically, they focus on the optimal selection of the downlink transmission power between the baseband units (BBUs) and mobile users, such that they reach the desired user QoS without exceeding the maximum capacity of the fronthaul link. At the same time, the authors emphasize that their algorithms are faster than other state-of-the-art algorithms, thus enabling their use in a more dynamic environment. Nonetheless, they express their convergence time only in terms of iterations, so an actual estimation of the adaptation time is not provided.

The implementation of CoMP over Cloud-RAN is also addressed in [115]. Its contribution on selecting and changing CoMP parameters in a fast manner allows the network to better support changes, thus making it more flexible. The authors focus on the optimal clustering of users and beamforming, such that a QoS-related utility function is maximized. In order to solve the optimization problem efficiently, they develop two algorithms that solve both problems suboptimally with relaxed constraints. Since the re-selection of parameters happens every scheduling slot, the adaptation time should be less than 1 millisecond.

The combination of CoMP and Cloud-RAN is tackled once again in [116], with a different objective in mind. This time, the authors present two low-complexity algorithms to select the optimal allocation of computational resources, RRH selection, and the beamforming in order to minimize the power consumption for Cloud-RAN. The fast adaptation of BBU and RRH parameters clearly improves the flexibility of the network. The work confirms the importance of jointly optimizing the power usage of the BBUs and the RRHs and proposes a cross-layer optimization scheme. The computation time required for the algorithms to converge, i.e., the adaptation time, is not presented, but the authors mention that real-time operation is still not possible in realistic scenarios.

Although the implementation of CoMP is a good opportunity to target flexible parameter configuration, there are other functions in a mobile network with configurable parameters. For example, Wang et al. focus on the scheduling parameters in [117]. They present a two-level scheduling algorithm for Cloud-RAN, in which scheduling parameters are dynamically configured to adapt to changes. These changes are usually traffic variations and user mobility, which could threaten compliance with computational or delay requirements. In response to this, the scheduler is designed to schedule users in a

wireless network while taking into account both requirements. The optimal beamforming parameters, modulation and coding schemes (MCS) and data rates are selected in a way that the system power is minimized and the required QoSs for the users are fulfilled.

Finally, FlexRAN [112], which is presented in the previous section, also features parameter configuration in virtualized control functions. Namely, it supports flexible reconfiguration of the policies (i.e., sets of parameters) that rule the control functions. In order to enable such reconfigurations, the authors design specific messages within the FlexRAN protocol that connects the central controller with distributed FlexRAN agents. By using this protocol, the parameters of the virtualized control functions can be updated at runtime in response to changes in the network.

In summary, parameter configuration in NFV is particularly important in virtualized radio access networks. More specifically, it is popular among those solutions including Cloud-RAN and CoMP ([114]–[116]), although the presence of CoMP is not mandated ([112], [117]).

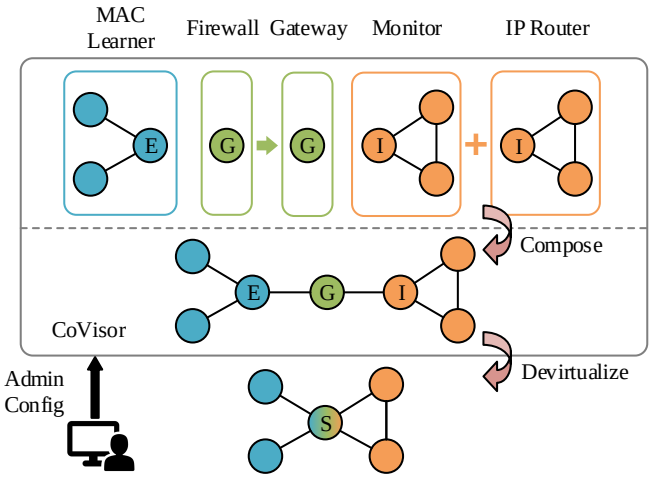


Fig. 6. In CoVisor [121], the policies from different applications are incrementally combined with the order specified by network administrator. Thereafter the combined policies are translated into flow rules, with the information of physical to virtual switch mapping.

4) Parameter Configuration in NV: Virtual networks, which are embedded in physical nodes and links, are logically isolated from each other. Every virtual network has its own policies and parameter settings, which can be updated by tenants or network providers. We analyze the flexibility of changing virtual network parameters in the following publications.

Bhatia et al. propose Trellis as a platform to provision virtual networks on commodity hardware in a flexible manner [118]. They criticize that container-based virtualization is inflexible, because all virtual hosts share the same data structures in the OS kernel, which potentially limits the diversity of virtual networks. As an improvement, Trellis can provide the possibility for tenants to customize their own IP network stack, such as parameters in congestion control and other algorithms. In the meantime, conflicting parameters of different virtual networks do not impact on the substrate's stability.

Gatekeeper [119] studies the problem of network virtualization in data center networks and targets virtual network performance isolation as the main design goal. The parameters are defined as the minimum guaranteed and the maximal allowed data rate among VMs of a network slice. Tenants are endowed the flexibility to set the bandwidth parameters, and at the same time, data center operators could still ensure the effective usage of underlying resources. To achieve such goals, tenant VMs are placed close to each other in the topology and built-in monitors report periodically traffic status to help control the transmission rate of VMs.

FlowVisor [120], as one of the first proposals of an SDN network hypervisor, virtualizes SDN networks on top of an SDN-based physical network. In the form of user defined policies, the parameters of each tenant virtual network slice can be flexibly supported and extended in the hypervisor. The policies in turn define the allocation of network resources. Evaluation suggests that such switch-level virtualization allows isolation of various traffic, and still achieves hardware forwarding latency.

The policy configuration of virtual SDN under hybrid network control is studied in [121]. The authors propose CoVisor (illustrated in Fig. 6) to process policies from multiple applications of different programming languages and from different controller platforms in a flexible manner. Each application, with its own virtual network view, updates the policies in response to different network events. CoVisor incrementally compiles such policies into a single composed one and sets up flow rules afterwards, which is several magnitude faster than a naive implementation. Therefore, the design could potentially adapt within a brief time upon network changes.

As expected, we observe that parameter configuration in NV mainly addresses the policies of the different network slices ([119]–[121]). However, other parameters of virtual networks, such as those of congestion control algorithms, can be also flexibly adapted by tenants [119].

B. Locate Functions

1) Function Placement in NFV: When functions are softwarized, they can be installed in any hardware that supports virtualization, regardless of vendors and models. Based on this premise, we may locate functions to achieve optimal performance metrics. For example, functions can be placed closer to traffic hot spots to avoid potential performance degradation that comes from increased buffer congestion level. Besides, migrating functions to unaffected area when node or link goes down ensures minor service interruption.

vConductor [122] is an advanced NFV management solution which manages multiple physical domains in a flexible and

automatic way. The authors apply a resource scheduler to decide the placement of virtual network functions in the underlying physical infrastructure, as well as the amount of resources that should be reserved to guarantee the performance of the functions. With traditional WAN (Wide Area Network) infrastructure, vConductor provisions cloud services that would span several data centers. Therefore, end users may enjoy shorter response time when applying services from a data center within close proximity.

Similarly, [123] also targets NFV orchestration that supports automatic function placement and dynamic lifetime control of the functions. The authors specifically consider a virtual router as a network function, and the adaptation of virtual router locations embodies the flexibility in function placement. Given a continuous knowledge of physical resource usage from the system monitor, three placement strategies dynamically propose candidates of new locations of virtual routers. A bagging scheme then decides the best one from the candidate list. As a result, different virtual routers can be instantiated on different physical hosts depending on several factors, e.g., infrastructure metrics and virtual network metrics.

As mentioned in flow configuration in SDN, [99] integrates SDN with NFV to enable a flexible function deployment, supported by traffic routing. In order to host possible new functions in the future, the placement of network functions takes system capacity into consideration. The authors devise a heuristic that clusters flows into groups and for each group incrementally places functions demanded by the flows. Therefore, function placement represents the major flexibility aspect of this work.

In [124], Chang et al. consider the use case of an SDN network offering abundant functionalities. They notice that one controller hosting all functions is inflexible and the overall performance could potentially suffer. To solve this issue, they propose an architecture named Hydra that performs functional slicing. Function scaling separates control plane functions and therefore they could be located in different servers. Meanwhile, a communication-aware function placement algorithm places the separated functions under the constraint of convergence time. The function reconfiguration latency is regarded as the adaptation time and such latency should be extremely small (in the order of milliseconds) for real time applications.

NetFATE [125] considers the function placement problem from a broader perspective, which incorporates end-users as potential locations to place functions. When considering traffic emerging from end-users, shifting functions towards them could in principle reduce the forwarding latency, as well as reduce the number of network entities in the forwarding paths. According to the authors, such shift could also make the service management more flexible. Given the topology and traffic distribution, the mathematical model minimizes end-to-end delay and therefore increases QoE for the users. Afterwards, the orchestrator can orchestrate network resources, allocate functions and decide traffic paths accordingly.

The problem of placing and chaining functions in data centers is studied in [126], which shows the flexibility in function placement. It is expensive to perform the optical-electrical-optical conversions between the optical steering domain and pods. Therefore, the VNFs from the same service chain are placed in the same pod under resource constraints, to lower the total embedding cost. The authors formulate a binary integer programming problem and propose a heuristic that produces nearly-optimal solutions.

Regarding mobile networks, function placement is quite popular for the core network, and is also featured in some NFV-based architectures of the RAN. The possibility to implement softwarized core functions in off-the-shelf equipment is very attractive for mobile operators, since it is cost-effective and softwarized functions allow for easy reallocations. Besides, function virtualization in the RAN facilitates centralization of the processing and hence coordination among cells, but at the same time it poses difficulties for cells to meet delay requirements. This dilemma has fostered flexible designs of the RAN that can take the best of both worlds.

Basta et al. in [127] present four alternative core network architectures for the location of virtualized S-GW and P-GW functions. Three of those alternatives imply a splitting between centralized and distributed functions, which can be flexibly chosen to meet delay or data rate requirements. In their study, the authors investigate the possibility to dynamically move the different functions to seek the optimal performance. The authors argue that the more functions are virtualized, the less the overall cost for the operators.

In [128], the authors analyze the optimization problem of placing the S-GW and P-GW functions within a network topology, considering two different scenarios: virtualized and decomposed functions. When functions are virtualized, both the control and data planes of the functions are softwarized and moved to a data center (shown in Fig. 7). In case of decomposed functions, only the control plane is implemented in the data center, along with an SDN controller. In both cases, an optimization problem tries to find the best function placement in terms of network load that fulfills the delay requirements.

The authors of [129] also tackle the problem of optimally placing virtualized S-GWs and P-GWs, but with the extra objective of minimizing the number of S-GW reallocations that user mobility can cause. Since their proposal allows for live migrations of the virtualized gateways, it can flexibly adapt to changes in user mobility. This mobility implies a trade-off between delay and signaling. Whereas it is advisable for the P-GW to be close to the UEs in order to avoid unnecessary delays, it may also cause problems if the S-GW are too close to the users. As a remedy, the authors allow for these functions to be located in separated, federated clouds and solve the optimization problem with these two conflicting constraints.

In [130], Bagaa et al. present a problem description similar to the last one, but they consider the existence of multiple PDNs providing different services or applications. In this case, the authors consider that a flexible placement of P-GW functions is necessary. In order to do so, they present and solve the optimization problem of placing these functions and

assigning them to UEs according to the services that the users demand. For this study, the objectives are twofold: reducing the operator costs and providing high quality of service.

A formulation for the problem of optimally placing the whole LTE service chain of softwarized functions (MME, HSS, SGW and PGW) is proposed in [131]. This proposal is another example of flexibility based on the dynamic placement of core network functions. The objectives are minimizing costs and meeting latency bounds for both user and control planes. After solving the optimization problem, the authors find that time required for the calculation of the optimal solution in a realistic scenario is usually less than an hour, low enough to allow for function reallocations in response to traffic variations.

The 5G NORMA project [109], already presented in Sec. IV-A1, also includes a high-level description of function placement in its proposed architecture. In fact, its design includes flexible allocation of mobile functions between the central and edge clouds, depending on the service requirements. This time, the task of moving functions relies on another element of the control plane: the Software-Defined Mobile Network Orchestrator (SDM-O). The SDM-O is in charge of locating functions in the optimal geographical locations, so that the latency and bandwidth constraints are met by the appropriate function chain.

Besides flexible in flow configuration, KLEIN [107] can also locate mobile core functions in a flexible manner, according to the required latency and function service chains. A key difference of KLEIN with respect to other proposals is the decoupling of control and data plane functions. That is, control and data plane functions can be placed in different data centers, even if they process the same flows. This decoupling reduces the complexity of the optimization problem. As an example, KLEIN is able to find the optimal function placement and flow configuration for a network of 2000 cells and 50 billion devices in around 20 seconds.

Regarding the RAN, in [132] the authors present RANaaS, a Cloud-RAN-based network in which functions can be either centralized (located at the data center within the BBUs) or distributed (located near the RRHs). The authors claim that the ability to move these virtual functions makes their proposed network more flexible than conventional mobile networks. The main feature of RANaaS is that the split between centralized and distributed functions can be flexibly moved along the functionality stack. In many situations, neither a centralized nor a distributed approach would be optimal, but a combination between the two. In order to allow such a trade-off, the functions in RANaaS can be dynamically placed, such that some functions are centralized in the data center and some are distributed in the RRHs. The details about the implementation of such a flexible functional split are explained further in [133]. Nonetheless, neither of the two proposals addresses the additional cost of provisioning both the RRHs and the BBUs to run the same functions.

Mountaser et al. [134] further elaborate on the idea of a flexible functional split in the RAN. They do not focus on the abstract design of a flexible splitting as previous work does, but on the implementation and measurement of one splitting option. These measurements provide information about the relation among latency, splitting and cell load, which is very important not only for the network design, but for triggering function reallocation. The authors consider a scenario of MAC-PHY splitting and Ethernet-based fronthaul, and they conclude that such a scenario fulfills the delay requirements for a 5G network, and thus can be applied to real networks. This paper is a good example of the impact of a allegedly flexible design on the latency of the network. However, similarly to the previous proposal, they lack an analysis on the additional costs.

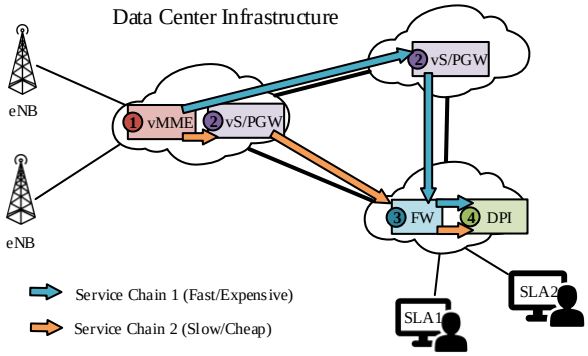


Fig. 7. Example of locating functions. The data center infrastructure consists of three interconnected data centers, hosting NFV 5G mobile core network functions, i.e., vMME and vS/PGW ([128], [168]), and legacy network functions, i.e., Firewall and NAT (Network Address Translation). Two service chains traverse the same order of functions, however with different latencies. Depending on the SLA that users choose, their traffic will be routed through different chains.

In [135], the authors build upon Cloud-RAN to propose FlexCRAN, an architecture that supports a flexible splitting between those RAN functions that are at the BBU and those located at Remote Radio Unit (RRU). Since the flexible RRU/BBU splitting consists of dynamically moving RAN functions up and down, FlexCRAN is more flexible than conventional Cloud-RAN, as it can better adapt to changes in delay requirements. For this, they derive a framework that contains all the components that are necessary to support that flexible splitting, such as interface functions, compression and synchronization units.

Finally, FlexRAN [112] also performs virtual function placement, as briefly mentioned in previous sections. In fact, the ability to dynamically place control functions is one of the most important flexibility aspects in FlexRAN. These functions can be dynamically allocated either in a central controller or in the eNodeB, with the intention of fulfilling the required delay constraints. In order to move these functions, the authors of FlexRAN have designed a mechanism to push code from the controller to the FlexRAN agents at the remote locations, which allows to effectively delegate functionality.

In summary, we observe that function placement in NFV is a very attractive aspect for all kinds of networks. For a generic wired network, the most pursued goals when dynamically placing functions are: reducing latency ([99], [123]–[125]), adapting the resource consumption [122], and reducing cost [126]. Regarding mobile networks, placing core or RAN functions to reduce the cost while meeting delay and data rate constraints is usually the main objective ([107], [109], [112], [127], [128], [131], [132], [134], [135]), although there are others such as reducing signaling overhead [129], or supporting multiple applications [130].

2) Function Placement in NV: The flexibility of placing functions, i.e., virtual nodes, is explored by the embedding process, i.e., Virtual Network Embedding (VNE) [139], [169]. A straightforward approach tackling VNE implies providing concise mappings in a centralized manner, with a full knowledge of the virtual network requests. Because network virtualization is applied mostly in wired networks, we provide an overview of the publications only in that domain.

Yu et al. in [138] observe that besides virtual node placement, virtual link assignment can also improve the embedding performance. They propose the "path split" mechanism, which allows provisioning virtual link with multiple physical links. In order to do this, they decompose physical link bandwidth into smaller resource blocks and then build up virtual links with the blocks. Consequently, the whole embedding process can happen in a more flexible manner and provides the potential to find solutions of VN requests in extreme cases.

The authors in [139] define formally the VNE problem with practical constraints, i.e. delay, routing and location requirements. An integer linear program optimizes the total cost of used substrate nodes and links, which benefits from the flexibility in function placement that NV offers. A benchmark set includes synthetic substrate topology and random virtual networks which arrive in a Poisson manner, and therefore produces promising comparison results.

[136] considers leveraging the flexibility of NV in function placement to guarantee reliability of virtual networks. Each virtual network, upon initialization, will be augmented with backup node and link resources. Without virtualization, such scheme may consume twice the amount of substrate resources. The backup resource pool provides the flexibility in provisioning recovered virtual networks under reduced cost.

The work in [137] moves one step forward, and enhances the flexibility to embed virtual networks in a pool of heterogeneous resources. Non-uniform substrates (e.g., servers and routers from various cloud service providers) are virtualized and treated fairly by the embedding engine. The authors design an Iterated Local Search algorithm to solve the optimization problem efficiently. Evaluation results show a great cost efficiency over a large number of VN requests, with minimum computation time.

Ludwig et al. [140] claim that while some parts of virtual network are fully specified by the tenants, e.g., the node and link locations, other parts may be flexible to decide by the providers, which can be exploited to improve the embedding of virtual networks. They develop an algorithm FlexMIP to leverage the specification flexibilities and are able to decrease the total resource cost under that specification. The work is further extended in [170], where they consider the time allowed to embed (i.e., schedule) the virtual network and they argue that time tolerance can bring in temporal flexibility.

In conclusion, we see that function placement in NV is mainly tackled by solutions addressing VNE problems. However, the objectives of these problems are rather diverse: minimal embedding cost [139], [140], maximal resource utilization ([138], [139]), enhanced reliability [136], or support of heterogeneous substrates [137].

C. Scale

1) Resource and Function Scaling in SDN: In SDN, the typical resource is the controller, as it implies limited processing power and I/O bandwidth. Therefore, proposals featuring this flexibility aspect in SDN wired networks usually tackle the management of controller resources.

Yao et al. assume that a controller is a network entity with a limited processing capacity and generalize the capacitated controller placement problem (CCPP) as an optimization problem [141]. Depending on the distribution of flows in the network, the minimum number of controllers that satisfies such traffic will be decided under the constraint controller capacity. Consequently, the load of maximum-load controller and the worst control latency are reduced at the same time. Because the problem is NP-hard, an efficient algorithm solves a series relaxed linear programing problems so as to gradually find the minimal number of required controllers and their locations.

The work [142] considers the controller placement problem in an SDN network consisting of a huge number of switches. Since it is not realistic that each node is a candidate for a controller location, the authors propose a hierarchical SDN controller deployment scheme. First, a fixed number of controller modules are placed to minimize worst-case control latency. Afterwards, a certain number of controllers in each module are instantiated (as in Fig. 8). For the first step, it shall be executed only if the topology changes, whereas the second step shall run more frequently to adapt to the traffic, thus increase system's flexibility. An algorithm called DyFlow is suggested which periodically estimates the number of packet-in messages from switches and activates only the required number of controllers.

Other than modeling and optimization, there are also works that prototype the scaling of controllers. ElastiCon [143] is a distributed controller architecture in which a set of controllers constitute a pool and its size could either grow or shrink depending on the traffic conditions. A monitor module collects load measurements at all active controllers and adjusts the number of controllers (adaptation). The authors treat packet loss and control channel interruption as cost of adaptation,

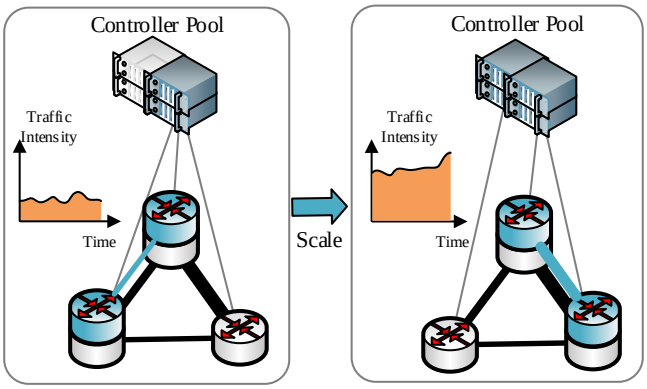


Fig. 8. Example of scale. The virtual SDN network experiences a dramatic increase of traffic and scaling is triggered to better accommodate the traffic. On the one hand, the mapping of virtual links moves to the substrate link with higher bandwidth, meaning topology adaptation. On the other hand, the controller pool assigns more resources to this tenant network, in order to handle the potentially rising number of OpenFlow messages.

and suggest a four-phase switch migration protocol to reduce the cost. The prototype evaluation suggests a much more flat response time curve with a changing packet-in rate.

Resource scaling, or resource allocation, is also a critical aspect of mobile networks. Apart from the computational and bandwidth resources that are usual in the wired domain, a RAN needs to manage additional resources such as spectrum or transmission power. In general, proposals in SDN featuring resource scaling are not as common as those in NFV and NV.

The most relevant example of resource scaling in an SDN-based mobile network is SoftRAN [144], a software defined architecture for the RAN, in which scheduling decisions are handled by a centralized controller. The controller is able to make better scheduling decisions since it manages all the resources. This increases network flexibility, as it increases the ability of the network to face changes in users distribution or channel quality. The reason is that the controller knows the users that are attached to all eNodeBs and their mutual interference, therefore it can allocate the demanded resources by using a time-frequency-eNodeB grid, which allows for less interference and more efficient use of resources.

To sum up, we observe that resource and function scaling in SDN may appear in two different ways. On the one hand, the controller itself can be considered as the function, or the resource, to be scaled ([141]–[143]). On the other hand, the separation between the control and the data plane may provide additional benefits of managing the rest of the resources in the network [144].

2) Resource and Function Scaling in NFV: The resource associated with a network function is intuitively the resource we would allocate for a piece of software, i.e. CPU cycles, I/O bandwidth, memory and hard disk size, etc. Following are the proposals that deal with the flexible scaling of NFV resources.

Carella et al. in [145] present an Autoscaling Engine (AE) to automate the process of NFV resource scaling. The implementation builds on top of Open Baton [171] and can be integrated in the ETSI NFV information model. Performance evaluation shows that in an IP Multimedia Subsystem (IMS) scenario, the scaling in procedure takes in total 513 ms. When CPU load goes beyond the threshold and scaling out is triggered, only 0.2% time of total procedure is spent in detection and decision-making, whereas 99.8% time is spent in deploying new function instances. The fast scaling process supports the flexible resource allocation to a great extent.

Data centers, as typical NFV infrastructure, provides both physical and virtual resources and compared with legacy cloud computing environment, resource management for NFV is more complex. In [146], the MORSA framework incorporates an NFV infrastructure resource filter and a resource scheduler to achieve flexible resource management in data center. Firstly the filter selects potential resources that could meet the requirements, e.g. hardware requirements and QoS levels. The scheduler then decides the best combination of resources and stakeholder policies. Because various stakeholders, i.e., end-user, data center operators and telecommunication operators, may have scheduling policies with contradictory objectives, MORSA leverages a genetic algorithm to fetch a list of Pareto optimal solutions, from which operators can freely choose according to their preference.

If we consider virtual machines in data centers as a bundle of resources, dynamic allocation during runtime enables workloads balancing across servers, racks and even data centers. The process of re-allocation improves the system's flexibility of resource scaling to a larger extent. VMs that are involved in the same task could show inherent dependency, therefore it is not practical to migrate a VM to a less overloaded server without checking the interconnection with other VMs. AppAware [147] is a novel scheme addressing such problem, which incorporates the information of VM dependencies as well as network topology. Evaluation results suggest more than 80 % network traffic decrease of the proposed scheme, compared with the state of the art.

Williams et al. study data center virtualization and argue that users should be assigned the title to flexibly control the virtualized resources, which consists of CPU, memory and hard disk, in data center [148]. They introduce Xen-Blanket to homogenize distinct cloud infrastructures, thus allowing live VM migration between enterprise cloud and public cloud. The actual migration time, however, is not clearly demonstrated. Besides, an elaborate coordination mechanism enables over-subscription of physical resources, which in turn increases the total revenue.

We mentioned in the previous section that resource scaling is very important in mobile networks, specially in the RAN. Furthermore, a technology like NFV facilitates the management of resources. Owing to this, it is not difficult to find proposals for an NFV-based flexible mobile network featuring resource scaling as their main flexibility aspect.

The most relevant example of flexible resource scaling in

an NFV-based architecture is Cloud-RAN [113]. Although it has been used by many researchers as a basis to develop multiple flexibility aspects, the main aspect of unflavored Cloud-RAN relies on the possibility of dynamically allocating the computational resources that the BBUs need. As it was previously explained, the basic idea of Cloud-RAN is to virtualize and pool the baseband processing of the eNodeBs into a centralized data center, whereas the remote locations keep only the electronic parts that are required to transmit a RF signal. This strategy allows for enhanced coordination and saves costs to the operators.

Based on the original idea of Cloud-RAN, [149] investigates further the flexible allocation of computational resources among cells in the BBU pool. The authors present this allocation as an optimization problem, which is solved by means of a Heuristic Simulated Annealing (HSA) algorithm. The simulations performed confirm that computational complexity of the HSA algorithm is linear, which enables its use when the number of cells in the BBU pool is large.

Moreover, all the proposals that combine CoMP and Cloud-RAN can be considered to be flexible at resource scaling, since they need to dynamically allocate power, time or CPU usage. For instance, in [114], the authors develop an algorithm to implement CoMP over Cloud-RAN. They focus on flexibly selecting the optimal allocation of the downlink transmission power between the baseband units (BBUs) and mobile users. Although the main focus of this algorithm is to dynamically select CoMP parameters, the selection of the transmission power can be regarded as a resource allocation problem, since the power is limited. The same can be applied to [115], [116], and [117].

Apart from the previous examples, there are a number of high level proposals of flexible mobile networks that include the resource scaling as a main aspect. In [150], the authors propose a cloud based platform for mobile network. In their design, the cloud should be able to dynamically allocate resources for the BBU when they are demanded. In [151], the authors emphasize the importance of a flexible resource allocation in the RAN as well as in the data center in the core network. Finally, in [152], the authors envision a Management and Orchestration (MANO) system that dynamically allocates the required computational resources for the virtual functions that are present in the control part of the network.

In summary, we conclude that resource and function scaling in NFV networks is a frequently exploited aspect, as the virtualization needs to deal with a variety of resources. More specifically, we identify three types of resources that an NFV network can flexibly scale: computational resources used by the virtual functions ([113], [145], [146], [148], [149]), virtual machines themselves ([147]), radio resources ([114], [115], [117]), and abstract resources in high-level designs ([150]–[152]).

3) Resource and Function Scaling in NV: VNE techniques tailor the allocated link bandwidth according to actual virtual link usage, and thus contributes to resource scaling of embedded virtual networks. [153] proposes a dynamic adaptive virtual network resource allocation algorithm Adaptive-VNE, which achieves flexible bandwidth reservation. Since the already embedded virtual links would normally not occupy the full bandwidth that they require, the unused bandwidth will be released and assigned to later new incoming requests. A monitoring module predicts an upper-bound of virtual link usage rate. Compared with static bandwidth allocation schemes, Adaptive-VNE maximizes bandwidth utilization and minimizes the bottleneck rate of virtual links.

He et al. study the problem of resource adaptation of virtual networks when facing multiple traffic classes [154]. They present DaVinci: an architecture that optimizes the aggregate utility of all virtual networks and can flexibly adapt to traffic variations. Every 10 seconds, it checks current link mappings and re-balances the bandwidth among virtual networks. Numerical experiments suggest that the adaptation to traffic shifts takes constant step sizes, however the actual time magnitude is not clarified.

Based on the well-studied multi-commodity flow problem (MFP), [155] tackles bandwidth allocation in network virtualization. The infeasibility of the optimization problem normally comes from substrate links that do not have enough capacity (i.e., bottleneck links). Therefore, granular allocation of link bandwidth, as well as adjustment of reserved bandwidth of bottleneck links, are jointly applied to enable flexibility in bandwidth scaling.

Virtualized SDN, as its name suggests, combines SDN and NV, and has emerged to an important research topic. Blenk et al. argue that the dynamics of networks not only reside at the physical infrastructure, but also at the controllers [156]. Accordingly, they introduce Hyperflex (i.e., a hypervisor) to isolate the control planes of different virtual SDN networks and thus flexibly manages substrate resources. The hypervisor forwards control messages coming from a tenant controller to the substrate switches that belong to the tenant. Both CPU and network bandwidth are treated as resources and are allocated accordingly with respect to a real time traffic scenario.

In data centers, networking bandwidth proves to be a scarce resource, for which various tenant networks compete. QJUMP [157] is proposed to tackle the recurring network interference, caused by virtual networks owned by different data center applications. The idea is to couple priority values and rate-limits. It guarantees high priorities by allowing data packets "jump-the-queue" over other packets with lower priorities. Therefore, bandwidth resource of each virtual network can be flexibly arranged according to user's demand. As a whole, QJUMP does not sacrifice transmission of high throughput applications; however, provides bounded latency for latency-sensitive applications.

Regarding mobile networks, there is a clear distinction between NV at the core network and at the RAN. Currently, network virtualization at the core network is just one application of network virtualization in the wired domain. Although there are proposals tackling the virtualization of a

mobile core network, few of them have specifically addressed any kind of resource scaling, to the best of our knowledge. Nevertheless, virtualization of the RAN resources does require special attention, due to the particular characteristics of this network.

SoftAir [106], which is already mentioned in Sec. IV-A1, includes a detailed study on the allocation of core and access resources for virtual slices of the mobile network. The authors propose to flexibly assign high- and low-level resources by using three different hypervisors. A network hypervisor is in charge of allocating high-level resources, such as the wireless spectrum, wireless infrastructure resources, and radio access technology options. A wireless hypervisor manages the low-level wireless resources, that is, the scheduling. Finally, a switch hypervisor assigns the low-level switch resources, i.e., the appropriate bandwidths for the different flows or virtual slices in every switch.

In [158], the authors present a two-level algorithm for allocating resources in a sliced RAN. Since this allocation is done in an adaptive manner, the network can flexibly cope with variations in the traffic required by the users. The first level of the algorithm schedules abstract resources within each slice, whereas the second level schedules actual resources to the slices. By using this scheduling strategy, the network operator is able to divide the RAN into virtual RANs, which can be rented to different tenants.

The authors of [159] propose a framework for RAN sharing, which abstracts real resources into virtual resources to be used by tenants. As a result, the flexibility of the RAN in terms of a more adaptive resource scaling is improved. Their key contribution relies on that they provide application-level services with guaranteed QoS, such that the scheduling of the slices takes into account the required QoS. With this cross-layer slicing, the network is able to allocate resources to those that need them the most, even if they belong to different tenants.

Finally, in [160], the authors present a scheme for resource reservation in the RAN that can flexibly adapt to the different traffic loads of each tenant. With such a strategy, this scheme provides better flexibility than previous alternatives of RAN virtualization. This is accomplished by using *partial resource reservation*, that is, a minimum slice of resources is guaranteed to each tenant, while the remaining resources can be shared by all of them. A custom scheduler is designed with the objective of allocating these common resources, in accordance with the amount of resources required by each tenant. Therefore, the share of resources actually used by any tenant at any time is dependent on their traffic loads.

To sum up, resource scaling in NV mainly applies to three resources: link bandwidth ([106], [153]–[157]), CPU usage [156], and radio resources ([106], [158]–[160]). Flexible solutions adapt these resources dynamically according to the requirements of the supported slices.

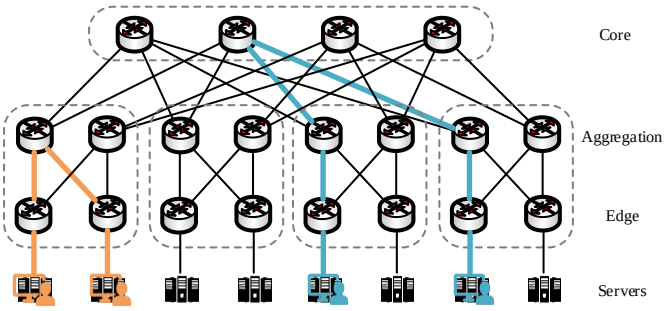


Fig. 9. A Fat-tree topology with $k = 4$, which illustrates a tree-like structure. Three layers of switches, i.e. core, aggregation and edge, enable multiple paths between different servers. A virtual environment (request), which consists of two VMs and a virtual link with a certain bandwidth, is embedded in the topology. As other requests come in and leave, the mapping of this request could be adapted from inside the same pod (orange) to stretching over different pods (blue), so as to ensure better resource utilization.

4) Topology Adaptation in NV: Another potential of NV is to change the mapping, which maps from virtual links/nodes to physical ones [172]. This helps to accept more virtual networks and to potentially gain more revenue. In other words, unlike changing the bandwidth assigned to a certain virtual link (as resource scaling would normally do), we embed it to another substrate link which possesses more capacity. Fig. 9 illustrates such an adaptation. As virtual network requests dynamically arrive and leave, topology adaptation achieves a more balanced resource usage distribution.

[138] introduces a path migration mechanism, which periodically monitors link usage information and forces new optimization of link mapping. It is an obvious indicator of flexibility in topology adaptation. The path migration time is treated as the adaptation time, and the service disruption during path migration is the adaptation cost. To minimize the adaptation time, node mapping is not dealt with; nevertheless, enabling node migration could provide even higher flexibility in adapting topology. Simulation results show in total an 80% (65% comes from path split and 15% comes from path migration) increase of revenue compared with a baseline algorithm.

[161] provides a greedy approach for network embedding. Fajjari et al. observe that most virtual network embedding rejections are due to bottleneck substrate links, which could be alleviated to accept more requests. The flexible topology adaptation comes from the reconfiguration of the embedded VNs, whose goal is to minimize the number of overloaded substrate links. In the meantime, the cost of reconfiguration in terms of service interruption duration is considered as the second minimization target. The cost is modeled as the weighted sum of migrated virtual nodes and links. The proposed heuristic reduces the rejection rate of VN requests by at least 83%.

Butt et al. [162] applies a new perspective to online re-optimizing the embedding. After a VN request is rejected, it recognizes the virtual link and node that cause such rejection. Next, the embedded links and nodes are reallocated to balance

the overall resource usage, leveraging the flexibility of embedding. By doing this, it can potentially accept large VN request by making more room of available resource. The embedding cost increase is considered in their evaluation, which is defined as the additional cost to re-embed VNs. The increase in cost ranges up to 250%, whereas the acceptance ratio improvement varies from 7% to 77%.

[163] studies the problem of recovering VNs affected by a substrate node failure, and proposes a reactive mechanism to get rid of inefficient pre-allocation of backup resources. The flexible adaptation of topology helps to achieve high reliability of VNs. Upon a single node failure happens, the proposed mathematical model maximizes the number of recovered virtual links across all the affected VNs and at the same time minimizes total bandwidth required for recovery.

Other than designing algorithms that output dynamic optimized topology mapping, there are also proposals that enable the flexible migration of virtual nodes and links from an implementation point of view. The authors in [164] propose LIME that migrates virtual machines together with the associated network and management system. To achieve transparency to running applications, LIME first copies data-plane state to new switches, and then migrates VMs in an incremental manner. During the migration, the states of both networks are synchronized to avoid inconsistent behaviors and interruption of applications.

The work in [165] moves one step further and tackles live migration of virtual SDNs. The migration, as the authors claim it, should be transparent to tenant controller and end-host applications. To reduce packet loss, clones of all or part of the virtual switch need to be created and afterwards synchronized to keep a consistent view of the virtual network. Obviously, synchronization prolongs the total migration time; however, prototype evaluation shows an acceptable migration time, which is around 0.2 seconds.

In summary, topology adaptation in NV is considered by the research community in two separate ways. On the one hand, some publications propose algorithms to adapt the topology based on changes in the link usages ([138], [161], [162]), or failure recovery [163]. On the other hand, other publications provide the technical details which allow for the topology adaptation: for a generic virtual network [164], or for virtual SDN networks [165].

D. Summary and Insights

In this section, we classify publications according to their flexibility aspect(s). In order to enhance the comprehension of this classification, we provide here a short summary of the whole section. A more complete analysis of observations and lessons learnt can be found in Sec. VI.

We first analyze publications which feature configuration adaptation. Unsurprisingly, we see that flow configuration is the main exploited aspect in SDN networks, as it focuses on creating, deleting and re-configuring flows within the network ([91]–[109]). In NFV networks, the most popular aspects are function and parameter configuration. We see that function configuration, which according to our definition implies completely changing the function operation, is more popular among mobile networks ([103], [104], [108], [110]–[113]). Parameter configuration is frequent in NFV and NV networks, specially for changing radio parameters ([112], [114]–[117]) and virtualization policies ([118]–[121]), respectively.

Regarding function placement, there are two possible realizations in the state of the art. The most straightforward realization is that of NFV networks, in which software functions (such as firewalls, load balancers, or LTE core functions) are moved from one possible location to another ([99], [107], [109], [112], [122]–[135]). Alternatively, this aspect is featured in some NV networks, when the problem is the location of the nodes of the virtual networks ([136]–[140]).

With respect to network scaling, it can be applied to all three technologies: SDN, NFV, and NV, although the implementation details are rather different. In SDN, the network controller itself can be considered as a type of resource that is susceptible to be scaled ([141]–[143]), or its presence can help to better manage other resources [144]. In NFV and NV, it is common that physical resources such as bandwidth or CPU usage are allocated dynamically among the virtual functions ([106], [113]–[117], [145]–[160]). Finally, topology adaptation is only possible in virtual networks, where it is indeed an important research topic ([138], [161]–[165]).

V. Classification According to Network Domains and Planes

In the last section, we perform a detailed survey of the state of the art on flexible networks and classify various works according to their respective focused flexibility aspects. In this section, we analyze the same publications from other viewpoints, in particular network domains and network planes. We do this to illustrate the impact of flexibility for different system types and the newly emerging split of data and control planes with SDN.

A. Network Domains

We primarily focus on three network domains in this section, i.e., Wide Area Network (WAN) and Mobile Core Network (MCN), access networks, and data center networks. Fig. 10 gives an overview of the classification according to network domains, where with each domain we associate the flexibility aspects that are primarily related to it according to our survey.

1) WAN and MCN: WANs typically span large geographical areas and offer high-speed data exchange between subnetworks. MCNs are a kind of WAN providing various services to mobile end-users, which are connected to MCN by Radio Access Networks (RAN). We find all three technologies of softwarized networks applied to WANs and MCNs therefore related proposals cover most of the flexibility aspects.

Flow configuration is the flexibility aspect that is featured most frequently in proposals. This is a consequence of the

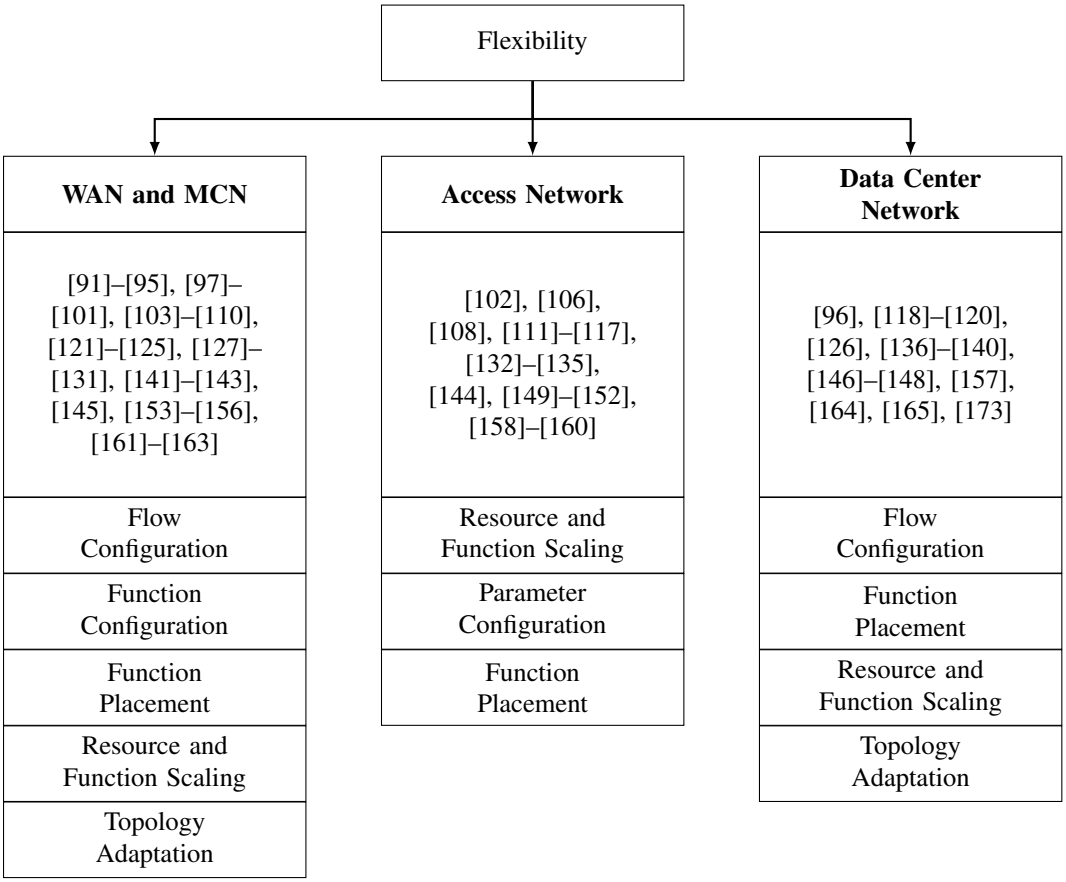


Fig. 10. Classification according to network domains.

great popularity of applying SDN in WAN and MCN. Examples include modeling and optimization in [91], [92], [99], routing algorithm design in [93] and the architecture proposals in [94], [97], [98], [100], [101]. We can also observe this in MCN papers, in which SDN is deemed to be a main driver of future 5G networks. Examples of this kind of proposals are CellSDN [103], SoftCell [104], MobileFlow [105], SoftAir [106], KLEIN [107], and SoftNet [108].

The second most popular aspect in WAN and MCN is *function placement*. Here, NFV is often regarded as a necessary complement of SDN. Examples range from use case feasibility studies ([101], [127], [129]–[131]) to implemented frameworks such as Hydra [124], vConductor [122], NetFate [125] and [99], [123]. Besides, *function configuration* is also often featured by flexible MCNs. This arises with the combination of NFV and SDN in core network proposals, therefore the examples for this aspect are similar to those of the previous one: CellSDN [103], SoftCell [104], and SoftNet [108].

NV is a main enabler for the remaining two major flexibility aspects, i.e. *resource and function scaling* and *topology adaptation*. For the former, virtual network embedding algorithms such as [138], [153]–[155] provide various solutions. Moreover, there are some proposals tackling the dynamic allocation of computational resources for mobile core functions, such as [106]. HyperFlex [156] investigates resource scaling of SDN control plane. Meanwhile, proposals, such as ([138], [161], [162], [174]), provide insights on embedding virtual networks more efficiently by adapting the embedded topology.

2) Access Networks: Access networks connect end-users with MCNs or backbone networks [175]. This definition implies that it has to include the required means to connect user devices to the rest of the network, by using special technologies and protocols, and to deal with user data, whose characteristics are often difficult to predict.

Not surprisingly, the most relevant flexibility aspect within Radio Access Networks is *resource and function scaling*, with a focus on resource scaling. This is due to the considerable number of wireless resources that have to be allocated in a flexible way, such as spectrum, time or transmission power. We can classify the access network proposals featuring resource scaling according to the resource they focus on the most: spectrum (SoftRAN [144], [158], [159], and [160]), computational resources (Cloud-RAN [113] and [116]), transmission power ([114]–[117]), and just a generic resource ([150], [151]).

Since it is quite related to resource scaling, parameter configuration plays also a significant role among flexible access networks. Most of the proposals featuring parameter configuration also include resource scaling, such as ([114], [115], [116], and [117]). Nonetheless, this combination is not always present, as in FlexRAN [112].

In addition, *function placement* in the radio access network has become important owing to the emergence of Cloud-RAN. Cloud-RAN enables the division between centralized and distributed units, which has triggered the research for the optimal functional splitting between RRHs and BBUs. Examples of proposals investigating the placement of functions between these units are [132]–[135], and FlexRAN [112].

Finally, there are two flexibility aspects that appear to be less popular, but still relevant in flexible access networks. These are *function configuration* ([111], and FlexRAN [112]), and *flow configuration* (of data plane and of control plane [102], [106]).

3) Data center networks: A data center network is the communication infrastructure in a data center, and it interconnects servers (physical machines) and storage devices in a data center facility. End-hosts in a data center network are VMs residing inside the servers, and the traffic behavior shows some unique patterns, i.e., (i) uneven distribution of traffic volumes among VMs, (ii) stable per-VM traffic at large time scale, (iii) weak correlation between rate and latency [176]. With load balancing and scalability issues in mind, data center topologies, e.g. Three-Tier, Clos [177], Fat-Tree [178], BCube [179] and Jellyfish [180], are designed, .

The special architecture of data center networks intuitively allows flexibility in *flow configuration*. For instance, SWF [96] handles the problem of dynamic routing in data centers with the support of SDN. Moving one step further, CheetahFlow [95] predicts frequent and heavy load transmission pairs and setup flow rules accordingly.

Flexibility regarding *function placement* can also be observed. Instead of deploying middleboxes, [127] proposes running mobile core functions in data center so as to save reconfiguration cost. Other works are demonstrated in ([122], [132]).

Moreover, since data centers provide resource sharing among multiple tenants or applications, efficient resource management is mandatory. With the help of softwarization techniques, data center solutions employ the flexibility aspect of *resource and function scaling*, supported by ([146]–[148], [157]). One the other hand, the *topology adaptation* aspect can be observed in ([138], [162]–[165]).

B. Network Control and Data Plane

Legacy communication networks are mostly vertically integrated. The part that decides how to process network traffic, i.e. control plane, and the part that decides how to perform forwarding actions, i.e. data plane, are rigidly compressed inside network devices. SDN separates the two network planes, and thus gives rise to higher flexibility in handling network's control logic and forwarding behaviors. Note that we do not differentiate control plane and management plane in this survey. Fig. 11 gives an overview of our classification according to network planes.

1) Data Plane: Data plane covers more than half of the papers that we have collected. *Flow configuration* tends to be the most popular aspect ([91]–[100]), as data plane traffic routing is a very basic task in data plane. In NV, virtual networks are embedded in data plane, we therefore attach all related flexibility aspects, i.e., *function placement* ([136]–[138], [173], [181]), *resource and function scaling* with a focus on resource scaling ([138], [153]–[155]), and *topology adaptation* ([138], [161], [162]).

Regarding wireless networks, a flexible data plane is also the main characteristic of most proposals. For instance, solutions dealing with CoMP ([114]–[116]), RAN virtualization ([103], [158]–[160]), and software-defined RAN ([144]) are focused primarily on the data plane.

2) Control Plane: In wired networks, SDN supports control plane flexibility. Intuitively, the number of controllers could vary according to the asynchronous message rate of the switches [141]–[143]. Regarding SDN virtualization, the control function, also known as hypervisor, can also be replaced and reconfigured on demand, e.g., Hydra [124], HyperFlex [101], [156], FlowVisor [120], LIME [165].

In wireless networks, a flexible control plane is usually linked to a flexible data plane, as in [102], [107], [109], or [132]. Nevertheless, in some cases the focus is set mainly on the control plane. Some examples of this are [104], [112] or [134].

C. Summary and insights

In this section, we reclassify the publications in the state of the art according to network domains and planes. Like in the previous section, we briefly summarize the highlights of this new classification here. We refer to Sec. VI for a more complete analysis of observations and lessons learnt.

First, we categorize the publications according to the network domain on which they focus: WAN and MCN, access networks, and data center networks. In WAN and MCN, we observe that flow configuration is the most exploited flexibility aspect ([91]–[94], [97]–[101], [103]–[108]), followed by function placement ([99], [101], [122]–[125], [127], [129]–[131]), and, to a lesser extent, function configuration ([103], [104], [108]), resource and function scaling ([106], [138], [153]–[155]), and topology adaptation ([138], [161], [162], [174]). In access networks, we identify resource scaling as the main aspect ([113]–[116], [116], [117], [144], [150], [151], [158]–[160]), followed by parameter configuration ([112], [114]–[117]) and function placement ([112], [132]–[135]). Conversely, in data center networks, there is not a single main flexibility aspect, as they typically combine the SDN, NFV and NV paradigms. As a consequence, the most frequently exploited aspects are flow configuration ([95], [96]), function placement ([122], [127], [132]), resource and function scaling ([146]–[148], [157]), and topology adaptation ([138], [162]–[165]).

Finally, we classify the publications according to the focused network plane: data or control. We realize that the majority of the publications mainly addresses the flexibility of the data plane ([91]–[100], [103], [114]–[116], [136]–[138], [138], [138], [144], [153]–[155], [158]–[162], [173], [181]), whereas that of the control plane is reserved for some SDN

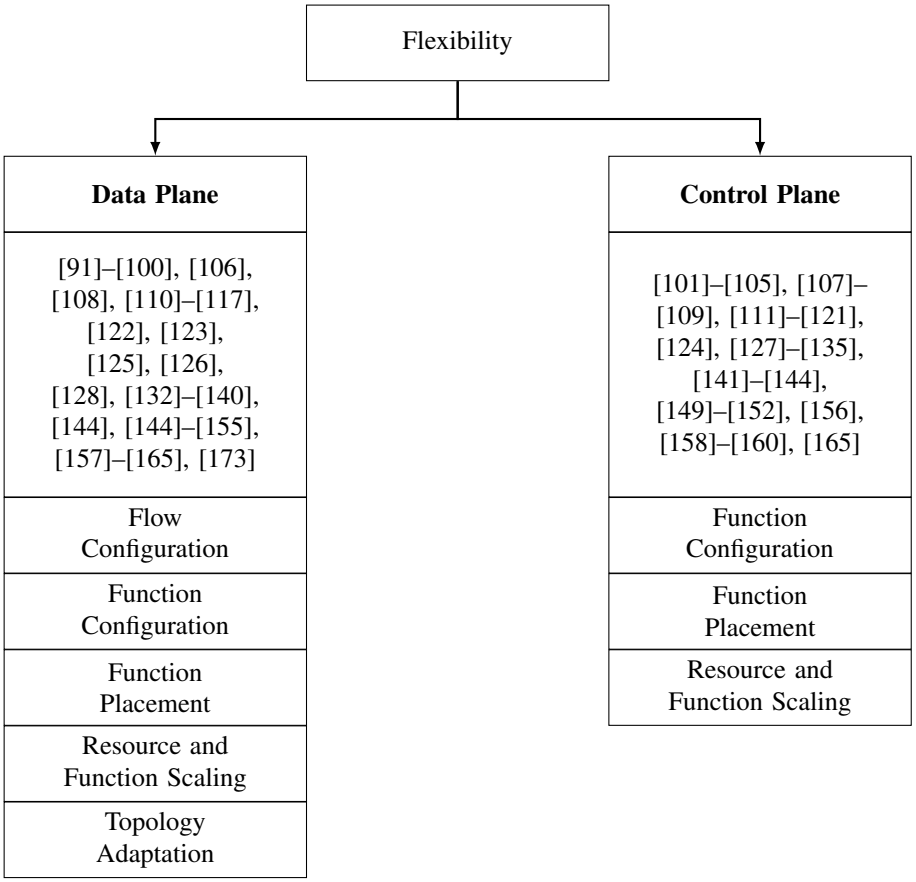


Fig. 11. Classification according to network planes.

solutions ([101], [102], [104], [107], [109], [112], [120], [124], [132], [134], [141]–[143], [156], [165]).

VI. Observations and Lessons Learnt

In this section, we compile observations and insights from the detailed analysis of the state of the art considering various flexibility aspects, as well as network domains and planes.

A. Presence of Flexibility in the Literature

After a thorough analysis, we can observe that the six flexibility aspects defined in Sec. III-B cover all surveyed publications. In other words, the flexibility in each publication could be categorized and decomposed into one or more of the aspects. This also includes those papers where flexibility is not claimed explicitly but identified from the content.

In some publications, only one flexibility aspect was exploited. In other cases, two or more aspects appear. Indeed, many flexible proposals do not stick to a single aspect but feature a combination of them. The reason is that usually a variation in a network element (e.g., a topology modification) implies a change in another element (e.g., new flows). Besides, technologies like NFV enable flexible manipulation of several aspects simultaneously. Therefore, multiple flexibility aspects can be leveraged when addressing a single request. We observe that frequent combinations of aspects are *parameter configuration* and *resource and function scaling* [114]–[117], *function placement* and *flow configuration* [99], [101], [107], [109], and *function configuration* and *flow configuration* [103], [104], [108].

B. Requests for Flexibility

Flexibility comes as a response to changes of network requirements. In Sec. III-A, we define those as *requests*. From our analysis of the publications included in this survey, we can draw some conclusions about the various sources of requests.

We observe that, in softwarized networks, there are at least five main sources of requests: traffic variation, user mobility, network upgrades, network lease and failure mitigation. In Table IV, we classify the surveyed proposals according to their considered source of requests. Depending on their main source of requests, proposals tend to exhibit similar characteristics, as explained in the following.

- **Traffic Variation:** It usually requires a flexible network to reconfigure flows, scale resources and even replace and configure functions. It is the most common source of requests for different types of networks.
- **User Mobility:** It typically requires the ability to configure flows, parameters, scale resources. It is one of the most common sources of requests in wireless networks.

TABLE IV
DEMONSTRATION OF VARIOUS REQUEST TYPES IN THE SURVEYED PAPERS

Request Source	Proposals
Traffic Variation	[91]–[97], [99]–[105], [107]–[113], [121]–[135], [138], [139], [141]–[145], [147], [150]–[154], [156], [161], [162], [164], [165]
User Mobility	[106], [111], [112], [114]–[117], [129], [132], [135], [144], [149]
Network Lease	[103], [118]–[120], [136], [137], [146], [148], [155], [157]–[160], [181]
Network Upgrades	[103], [104], [112]
Failure Mitigation	[94], [97], [98], [163]

- **Network Lease:** The network has to cope with changes coming from tenants' requirements, such as VM size, inter-connection bandwidth, and virtual topology, thus resource scaling and topology adaptation are the most exploited flexibility aspects in this case.
- **Network Upgrades:** The network operator could either enhance network performance (e.g., lower latency and higher throughput), or just employ new protocol/mechanism/solution in the system.
- **Failures Mitigation:** It entails the ability to reconfigure new service or restore network connectivity upon failures.

Although these sources of requests employ some flexibility aspects more frequently than others, there is no one-to-one correspondence between them. This means that the same request may be fulfilled by exploiting various aspects. For example, in order to respond to a change in the requirements that demands a lower latency constraint between an SDN controller and a switch, the network could either move the controller function to a location closer to the node or forward the control traffic through a shorter path. In this case, both *function placement* and *flow configuration* are valid solutions to the request of decreasing the control latency.

C. Adaptation Time

The support of requirement changes should come in a timely manner, i.e., the time spent to adapt the system ought to be considered. The ability to adapt rapidly is a necessary condition for flexibility, regardless of any other feature. For instance, a self-healing network could hardly be asserted as flexible if automatic link failure detection and repair is slower than manual intervention. surveyed.

After analyzing the surveyed papers, we observe that the adaptation time depends on technology, network domain, and use-case. Owing to this, we observe different values for the required adaptation time, ranging from minutes to milliseconds. At one end of the spectrum, virtual function placement in the mobile core network can take up to an hour [131], as the optimization problem is computationally hard. Faster adaptations are featured by SDN-based solutions, including flow path redirection, switch migration, and control plane reconfiguration. The goal of flow path redirection latency is in the order of tens of seconds, as the flow rule installation time can be significant due to traits of underlying hardware [98]. Migrating switch between two controllers should take less than 100 milliseconds [143], and it also depends on the OpenFlow packet rates. The control plane reconfiguration, i.e., control channel migration, should be in the order of tens of milliseconds [101] to avoid delays of OpenFlow packets.

Shorter adaptation times can be observed in other use-cases. In a real-time NFV scenario, function reconfiguration is expected to meet the deadline of 10 milliseconds, indicated by the heart-beat synchronization event [124]. Similarly, flexible proposals concerning the RAN often consider adaptation times in the order of milliseconds. For example, [111] states that the switching between FFR and DAS configurations must be done within 10 milliseconds. Another example is [133], in which a deadline of 3 milliseconds is considered for the function placement, based on the restrictions imposed by the Hybrid-ARQ. We can see from these examples that stringent target adaptation times are usually motivated by the presence of external deadlines.

On the other hand, some publications do not mention the target adaptation times while proposing their new algorithm, mechanism, or system architecture. For instance, [94] studies the link failure recovery on SDN-based clouds, where the recovery directly relates to the adaptation, but it does not explain how fast this recovery should be. Neither does [138], which discusses the impact of link migration on virtual network service interruption. Finally, [129] and [130] address the problem of virtual function placement in the MCN, but, as opposed to [131], they do not provide an estimation of the placement time. In these cases, it is assumed that the adaptation is done as fast as possible, but this is not enough for a proper definition of flexibility.

In conclusion, we observe that the adaptation time is indeed technology- and domain-specific. In addition, the adaptation time is still not paid attention to in some work, specially if no tight delay constraints are present. We therefore advocate keeping the time aspect in mind while designing flexible systems to indicate that the adaptation is indeed useful.

D. Cost of Flexibility

Cost-Effectiveness. Regarding cost, there are only a handful of proposals that leverage the concept of flexibility to achieve cost-effectiveness. In mobile networks, for instance, network

operators could reduce their cost in two directions. On the one hand, investment could be saved by virtualizing the network functions and operating them on cloud platforms (e.g., [111], [113], [114], [132], [135]). In fact, the authors of Cloud-RAN [113] envision 15% CAPEX and 50% OPEX savings when compared to previous less flexible RAN designs. On the other hand, slicing RAN resources (e.g., [159], [182], [183]) promotes high network utilization ratio via network leasing, and therefore potentially increases the total revenue. In [183], the authors expect savings from mobile network slicing up to 40% for OPEX and up to 15% for CAPEX.

Potential savings are not unique to mobile networks, but a similar trend can be observed also in the domain of WAN and data center networks. Network service providers will potentially see a reduction in OPEX, because SDN and NFV enable further optimization of network resource and surrounding operation model [184]. As a concrete example, [126] places network functions of a service chain in the same PoD and thus decreases the embedding cost of service chaining by eliminating frequent optical to electrical conversion. Besides, the use of resource pool, composing of virtual nodes and links, shrinks the cost of re-embedding [136], when virtual networks need to be recovered from infrastructure failure.

Implicit Cost Factors. Even though cost-effectiveness is targeted among surveyed papers, we can still observe that cost factors are overlooked among some others. For instance, SDN enables reconfiguration of flow paths on run-time and leveraging this advantage, proposals like [91]–[94], [96] manage to adapt the forwarding path according to the demand changes. However, if the reconfiguration takes place while packets forwarding is still happening, longer latency or packet drops are expected. To avoid this drawback, an elaborate coordination mechanism should be designed to ensure the correct transition from the old to the new forwarding rules. Such mechanism in turn may introduce additional signaling and leads to increasing the system's cost, which we refer to as implicit cost factors. Another intuitive example is the "over-provisioning" strategy. Proposals like [136] suggest providing more resources than what is actually needed to avoid frequent reconfiguration that may lead to system instability. The resource headroom could be seen as non-negligible cost at the start-up phase, however, its impact could also be diluted if we consider from a longer time span, as all later requirement changes would be satisfied without any violation of SLAs. Therefore, we strongly call for the attention of all cost factors involved while evaluating the network's flexibility.

Impact on Network Performance. Flexibility enhancements can also affect the performance of some network transactions, which has a clear impact on the cost. For instance, the network's latency is prone to be affected by such enhancements. A latency increase implies additional cost, since the network needs to reduce the number of supported users, compensate with over-provisioning, or face fines for service infringement. Owing to that, high latencies are in general avoided in flexible designs, but there are still some cases in which a flexible network comes at that additional cost. Among those, we can find proposals based on the Cloud-RAN architecture[113], which introduces delay in the link between the baseband and RF processing parts of the eNodeB, in return for lower hardware costs and better resource management.

On the other hand, in many cases a flexible network can perform better than an inflexible network, resulting in reduced costs. For example, efficient and optimized algorithm designs contribute to fast decision making ([96], [100], [114], [116], [124], [185]). Besides, better coordination of information exchange between functions [115] is also beneficial, if the information plays a key role in the system performance. Finally, in the context of failure recovery [94], [98], shorter time heralds less number of dropped packets upon link/node failures.

TABLE V
NUMBER DEFINED OF MATCH FIELDS AND ACTIONS IN DIFFERENT OPENFLOW VERSIONS

OpenFlow Version	1.0	1.1	1.2	1.3	1.4	1.5
# Match Fields	12	15	36	40	42	45
# Actions	4	6	7	7	7	9

E. Impact of Standards and Technology Realization

Standards. Hanseth et al. [186] explore the limitations of flexibility due to standardization in legacy communication networks. Even though network softwarization enables more flexibility, it still inherits such standards limitations. SDN, as a main technology of network softwarization, supports programmable flow forwarding based on packet headers. Its flexibility nevertheless has an apparent upper bound and the reasons are threefold. First, the supported match fields in the SDN de-facto protocol, i.e., OpenFlow, to filter out a target flow are still limited, even though the number has increased from only 12 match fields in v1.0 to 45 in v1.5 (shown in table V). Second, an OpenFlow switch has also some limitations in packet forwarding, a simple example being stateful operation. Only basic operations like set, copy, increment/decrement, pop/push are defined in the standard specification. Last, the new features brought in by new OpenFlow versions typically indicates minor changes (if not dramatic as from v1.0 to v1.1). Similarly, research on flexible solutions for mobile networks are often weighed down by the modifications of the standard that they target. Indeed, one can find that the ability of making the network flexible without a lot of modifications to the standard is presented as an attractive feature [107], [112].

Technology Realization. In some cases, the realization of flexible concepts poses problems that are difficult to predict at the design phase. As a result, the details of this realization may restrict the final flexibility of the network. In NFV, for instance, the implementation of computing hypervisors has a direct impact on virtualized network functions. Different virtualization techniques, i.e., full virtualization, para-virtualization

and binary translation [187], could incur various issues. Para-virtualization requires the kernel of the guest system to be aware of the virtualization and thus the guest system lacks the full support of instructions on the computing processor. Full virtualization, on the other hand, authorizes the guest system to operate without any modification, which offers more flexibility to VNFs. In conclusion, flexible proposals should carefully consider the implementation problems that may appear when bringing the design into fruition, as they can impact on flexibility.

F. Towards Other Indicators

Finally, we observe that under certain contexts, other networking performance indicators, such as QoS, security and resilience, are also influenced by flexibility.

QoS. Flexibility, in general, can help guarantee or even improve QoS. With a global QoS level and changing input traffic, flexible configuration of flow paths can help enforce the latency, bandwidth, jitter, and other requirements [104], [115]. Furthermore, function placement is exploited by several solutions to help meet the promised QoS constraints [109], [127], [130]. In the case of [109], even a dedicated QoS monitoring infrastructure is envisioned. Finally, resource scaling and parameter configuration are very useful means to achieve a required QoS, as shown in [114].

Security. Flexibility can pose either positive or negative influence on network security. For the former case, we can model network attacks as *requests*, so that defending against such attacks is the same as accommodating the requests. An example of this is [100], in which the flexibility in *flow configuration* improves network security. Conversely, the reconfigurability that is associated with flexibility can open the doors for new attacks. For instance, in [112] new security threats appear, due to the possibility of an attacker to exploit the function placement mechanism to alter the code of legitimate functions.

Resilience. Similar to security, the influence of flexibility on resilience can go in two directions. On the one hand, network failures can also be modeled as *requests* and treated accordingly. Concretely, we observe some potential failures that flexible softwarized networks would encounter and cope with, e.g., inactive SDN switch [94], [98], commodity server fault and link failure [136]. On the other hand, some flexibility-enabling technologies can decrease the resilience of the network. For instance, the central pooling of resources that enables resource scaling flexibility implies increased risks with respect to distributed solutions, as the resource pool is a single point of failure [113].

VII. Opportunities and Research Challenges

From the flexibility analysis and observations, we can infer that the goal of higher flexibility incurs both future opportunities and challenges for researchers. In this section, we start with highlighting the opportunities, i.e., how the flexibility can enable the goals specific to various network applications. Afterwards, we collect the potential research challenges, ranging from the buildup of the flexibility quantification framework to the application of automation and AI, and identify the needed steps to achieve a flexible network design. Table VI summarizes all the challenges and the potential methodology.

A. Opportunities Towards Flexible Network Applications

Considering flexibility can benefit greatly the design of future network applications. In this section, we demonstrate this benefit by introducing five network applications (i.e., enterprise network, smart grid, 5G, wireless sensor network and Internet of Things) and discussing the relation between the requirements of the applications and our derived flexibility aspects. With flexibility in mind, network designers can quantify the degree to which a certain design option can satisfy the requirements and therefore compare different design options.

Enterprise network. Managing an enterprise network used to be expensive and error-prone, because of a variety of running applications and protocols [188]. The high management cost derives from the rigidity of the middleboxes and the operation complexity to recognize and resolve the errors and anomalies in a timely manner. To address this issue, an enterprise network should be able to provide on-demand network services with software NFs [189], [190] (*function placement*, *function configuration*, and *resource and function scaling*). Besides, the network needs to be flexible in routing traffic flows (*flow configuration*) towards an optimal and seamless operation.

Smart grid. A smart grid is composed of several electrical, control, and electronic devices, and it provides power supply to end-users in an efficient and reliable manner [191]. The network supporting a smart grid should provide strict QoS guarantees and ultra-reliability. Hence, a smart grid network design needs to incorporate flexibility of recovering flows (*flow configuration*) in failure situations as well as flexibility of changing the network parameters (*parameter configuration*) to adapt to changing services and serve the demands of various use-cases.

Fifth Generation Mobile Communication (5G). The next-generation mobile network poses challenging requirements with respect to LTE, such as higher capacity, higher data rate, lower device-to-device latency, and consistent QoE provisioning [192]. These requirements have to be fulfilled while facing constant changes in user traffic, interference conditions, operator demands, and future standards. Thus, a 5G network design needs to offer flexibility in terms of intelligently routing fronthaul/backhaul traffic (*flow configuration*), switching between alternative RAN and core functions (*function configuration*), moving those functions to the most convenient locations (*function placement*), dynamically adapting function policies (*parameter configuration*), efficiently managing radio and computing resources [193] (*resource and function scaling*), and providing configurable slicing for virtual operators (*topology adaptation*). However, the ability of performing the actions above is not enough. For a flexible 5G network, such adaptation also needs to consider time constraints, which range from the order of microseconds for adapting radio resources

TABLE VI
SUMMARY OF RESEARCH CHALLENGES

Challenge	Potential Methodology/Solution	Comment
Flexibility measure	Ratio supported requests (=challenges) over the total number of requests can serve as a measure to compare different systems w.r.t. the same flexibility aspect.	-
Network architecture design	Optimization for a flexibility aspect as a special performance metric to derive insights towards a flexible architecture design.	-
Trade-off between over-provisioning & adaptation	Model all the cost factors, including CAPEX and OPEX, and analyze the trade-off with the achieved flexibility value in mind.	Different network systems may have different preferences.
Manage unpredictability & instability	Perform measurements on physical devices, collect performance issues, and consider the issues when designing flexible systems.	Hard to predict exact performance
Combine software & hardware	Integrate hardware acceleration for computation-intensive tasks, implement software network function with high efficiency and reliability, design hybrid resource management mechanism.	-
Algorithm Design and Artificial Intelligence	Design more efficient algorithms to address more complex optimization problems, and apply AI for faster decision making in network management.	-
Automation	The SDN controller implements the automation of traffic monitoring and flow path engineering. For NFV and NV, the resource manager reacts to request input and potential failures.	The full process of network service provisioning should be automated.
Benchmarking	Based on measurements, generate realistic datasets of different request sets that demand for flexibility.	-

[194], to the order of minutes for the modification of virtual slices [195].

Wireless Sensor Network (WSN). By means of sensors that can measure physical and environmental parameters, a WSN collects data over a certain area and forwards them to a central location for processing [196]. As a great constraint to WSNs, all forwarding nodes are supplied with limited supply of energy. WSNs should also be able to timely adapt to sudden events, change of interference conditions, and future updates. A flexible WSN should implement an efficient radio resource algorithm (*resource and function scaling*) for dynamic channel condition to save energy, as well as a multi-hop strategy for data collection [197] (*flow configuration*), especially when new sensors join and existing sensors die during the operation. In the meantime, the topology should adapt to various use-cases (*topology adaptation*) in the field such as tracking, moving components, environmental setups, etc.

Internet of Things (IoT). The future networking paradigm entails the inter-connection of nearly every device in our daily life. It is estimated that 20 billion to 50 billion devices will be connected to the Internet by the year 2020 [198]. Clearly, a network exploiting this paradigm needs to exhibit a design able to scale with the huge amount of devices, which varies over time and location, and handle the highly-variable traffic generated by the devices. Specifically, the flexible coordination of network resources can address the large amount of different service requirements originating from the heterogeneous devices [199] (*flow configuration*), and, by utilizing SDN and NFV, the resources for individual IoT use-cases can be flexibly allocated [200] as well as contribute to the global resource efficiency (*resource and function scaling*).

B. Flexibility Quantification Framework

1) Flexibility Measure: Flexibility is a desirable characteristic of a network. However, it is usually ambiguous how to quantify network flexibility and compare different network design choices. For this purpose, a quantitative flexibility measure is needed. Such measure could take into account three components, as explained in Section III: (i) definition of requests, (ii) definition of flexibility aspects, and (iii) adaptation time constraints.

One possible way is to measure flexibility as the ratio of the number of supported requests, that can be realized under an adaptation time constraint, over the number of all given requests. In our preliminary work we apply this methodology and draw useful insights by comparing different system design choices, e.g., for mobile core network functions placement [8] and for dynamic SDN control plane [7]. The dynamic SDN control plane, for instance, evaluates the question "is a distributed control plane more flexible?", which may look intuitive. However, it turns out that under some circumstances, a centralized control plane is also flexible enough to handle the varying traffic, while adding only relatively low cost overhead.

2) Requests and State Representation: The quantitative evaluation of the flexibility of a system could be a complex and challenging task. First, the full set of all possible requests that could change in a networking system should be defined,

whose cardinality can be infinite. For example, the data and control latency requirement is theoretically any real number that falls into the range of acceptable operation. Second, in case an adaptation is required, the state of the system also poses an impact. Consider the case of dynamic SDN controller placement for example as in [7]. The migration time of the SDN controller to the optimal location depends on the original location of the controller, i.e., current system state. Therefore, the representation of the request space, together with the system state, is one of the most important challenges to overcome.

3) System Comparison: Moving one step further, we throw out another interesting question: "is it possible to compare the flexibility of any two networks?". This question can be trivial if we examine two networks that are intended for the same type of requests. For example, consider two SDN WANs with the same topology and facing the same requests (new traffic distributions), but with a different number of controllers. The network that can accommodate more traffic distributions can be easily seen as more flexible. Moreover, we can even compare networks realized with different technologies, e.g., SDN-based and NFV-based MCN, provided that we have carefully designed a set of requests and a common adaptation time constraint.

The same question, however, becomes challenging if we generalize the comparison even more. It is not evident either how to compare flexibility of networks that face different requests, or that have different time constraints. Still consider the case of two SDN WANs, but this time they face different sets of traffic distributions. One can accommodate five out of five distributions, while the other is only able to accommodate eight out of ten distributions. If we opt for the absolute number of supported requests, the later one is more flexible. On the other hand, if we prefer the ratio of supported requests, the former one becomes more flexible. A difference in comparison methodology results in completely different conclusions. To this end, we need to think of how to get rid of such ambiguity, i.e., normalize the various factors, and thereafter perform comparisons that make perfect sense.

C. Flexible Network Architecture Design

Having a flexibility quantification framework will enable the design of future flexible network architectures, which is also a challenging task. The first step is request representation, i.e., to define all possible network requests that show both temporal and spatial changing behavior. Examples of such inputs are traffic, function (including type and location) and topology variation. Next the flexibility aspects should be modeled as network design optimization problems, which targets specific technology and concept details. The optimization objective is to maximize the number of supported requests, under the constraint of adaptation time, as well as the constraints that are inferred from other network indicators, e.g., data plane latency, signaling overhead, or bandwidth usage ratio. In the end, by analyzing solutions to the optimization problem, we could hopefully derive patterns and insights that attain a flexible architecture. One possible pattern would be that topologies with a specific property, such as high betweenness centrality, could accommodate more traffic variations without much need of reconfiguration. We would like to see more work on elaborating such a flexible network architecture design methodology.

D. Over-provisioning vs. Adaptation

As mentioned in Sec. III-A, the network may accommodate the requests by modifying its topology, functions, flows, and resources. Nonetheless, such modifications are not always required, as adaptation may be compensated with over-provisioning.

By assigning more resources than actually needed, the network would be able to address common challenges, such as increased traffic demands and more stringent latency requirement, without the need for any change in its state. This might be beneficial from a performance point of view, since adaptation may incur reconfiguration latency. Indeed, performing adaptations means higher risk of violating the time constraint, which is a severe problem in many communication networks. However, such a method obviously calls for higher CAPEX in contrast to a network with less resources that can dynamically adapt them. Conversely, the OPEX might be lower in an over-provisioned system, because the operation of the network would be less complex. However, as more resources need to be utilized, the related operational cost can also increase. To this end, a clear analysis of all OPEX components is needed to clarify its relationship with over-provisioning. In conclusion, any flexible design should carefully consider the actual benefits and costs of performing adaptation, in contrast to resource over-provisioning.

E. Unpredictability and Instability

The performance guarantees in flexible network systems are another important challenge. For instance, virtualized network functions are hosted in VMs, and the isolation of VMs guarantees the agreed service level between tenant and infrastructure operator. Indeed, a given fraction of every resource type, i.e., processor, memory and I/O bandwidth, is defined for each VM. However, as pointed out in [201] and [202], micro-architectural level resources such as the caches inside a processor cannot be divided and then dedicated to each VM. In other words, all running VMs share them in a competitive manner, which may result in cache contention. A CPU intensive VM would keep overwriting the cache, resulting in higher latency of retrieving data for other VMs. The exact performance of network functions therefore becomes harder to predict.

In addition, as softwarization introduces one more virtualization layer to the system, the errors of the physical layer, even tiny ones, will be propagated to the above virtualization layer and trigger instability of virtual networks [5]. For instance, a misconfigured routing protocol in the

substrate causes routing oscillation and as a result, virtual networks suffer from instable packet forwarding delays and probably packet losses.

F. Software and Hardware

There has been a long history of discussion about software and hardware, which also relates to flexible communication networks. We envision softwarization as the key enabler of flexibility, and in the meantime, we are also aware that compared with proprietary and rigid hardware network equipment, software solutions are not competitive in terms of absolute processing speed. To this end, the idea of leveraging commodity hardware for various network services is widely adopted both within academia and industry, e.g., P4 ([54]) that enables data plane programmability and white box switching ([203], [204]) that leverages switch commoditization. Software programs, as an effective supplement, not only implement network services, but also operate hardware resources and orchestrate the implemented services [205]. There are several available directions of improvement that we need to work on, namely (i) integrating hardware acceleration configuration, e.g., advanced memory read/write and I/O speedup, into the servers to enhance the packet processing capability; (ii) implementing network functions in software with high efficiency and reliability; (iii) designing sophisticated resource management mechanism to utilize hardware in an optimal manner.

G. Algorithm Design and Artificial Intelligence

Flexibility brings in new challenges to network optimization, since the problem space grows drastically with more degrees of freedom in the system choices. For example, the placement of virtualized network functions is harder than that of hardware middleboxes, because VNFs can be instantiated and distributed among more locations, i.e., servers and cloud platforms, than in case of middleboxes. Efficient algorithms as in [91], [92], [99], [115], [206] are needed to speed up the procedure. Since the time aspect plays a vital role in flexibility, if the system could make faster decisions, it will be able to handle more requests and therefore be more flexible. We envision machine learning as a promising approach to boost the process of decision making. Take the area of virtual network embedding or cluster assignment in data centers as an example. For virtual network embedding, a recurrent neural network learned from algorithm results to predict the outcome of future algorithm executions [207]. In data centers, a data-driven approach [208] performs admission control and even increases the embedding efficiency; for data center job scheduling, a reinforcement learning approach outperforms shortest job scheduling [167]. Generally, the idea to learn from algorithm data has been demonstrated for facility location problems, virtual network embedding, and the dynamic controller placement problem [209], [210]. In order to fully exploit the flexibilities of softwarized networks, we envision a high demand for such algorithmic concepts in the future, i.e., a high demand for conducting future work in this area.

H. Automation

Another challenge is that the complete process of statistics monitoring, decision making and adaptation execution is required to be fully automated for flexible softwarized networks. Indeed, autonomous networking has become a hot topic recently, showing a trend of less human intervention in network management. In SDN, the controller acts as the network brain that controls the underlying switches and their supported functionalities. We would like to see more contribution in controller implementation that handles automated network provisioning.

Similar trend of automation in demand-driven elastic management of infrastructure also happens in NFV. The tasks becomes even more complicated when facing heterogeneous hardware resources, which is exactly the case nowadays, as servers with different capabilities need to be treated equally and virtualized. According to [211], automatic NFV management should be able to: (i) estimate VNF capacity, (ii) compute virtualization and system overhead, (iii) determine the optimal resource configuration, (iv) evaluate different virtualization and hardware options, and (v) tune VNF implementation and performance. Researchers should pay attention to all five components when proposing an automation framework.

I. Benchmarking

For classical problems, e.g., Traveling Salesman Problem, Facility Location Problem and Time Scheduling Problem, common datasets have been proposed as benchmarks to compare those different algorithms. This is yet not the case for most of the problems occurred in flexible softwarized networks. One example is the study of dynamic control plane, which depends on the distribution of control plane traffic. However, uniform distribution, while mostly used, does not reflect the realistic property of traffic. Another example can be observed with the VNE problem in the area of NV. Concretely, the structure of VN requests has a considerable influence on the VNE algorithm performance. There is yet no common dataset of VN requests [173] in the community. The majority of VNE papers apply VNs that are randomly generated, without consistency in the number of nodes or the distribution of links.

In order to target this challenge, we advocate the necessity to have common practical datasets of each request types, e.g., traffic variation, user movement, VN request, etc., to evaluate the flexibility of different proposals. Of course, such datasets should be built upon comprehensive measurements with different use cases in mind.

VIII. Conclusion

Communication networks have to cope with frequent changes in user requirements, traffic distributions, service demands, and system anomalies. This has boosted research efforts towards designing flexible networks, in order to accommodate such changes in a timely manner. Such efforts have flourished in softwarized networks, that is, SDN, NFV, and NV, as they provide a high level of adaptability and reconfigurability. However, a common understanding of flexibility is missing in the networking literature. This hinders the comparison of different network designs, and also challenges developing even more flexible network concepts. In order to address this, we initiate the study of network flexibility and present this survey as a comprehensive analysis of flexibility in softwarized networks.

We first propose a definition of network flexibility, based on the common notion existing in the literature. We expose the multifaceted nature of flexibility by decomposing it into three categories and six flexibility aspects. We rely on a combination of those flexibility aspects and network technologies to analyze and classify the state of the art. Moreover, our classification also covers different network domains (such as wide area, mobile and data center networks) and control and date plane.

In the light of this classification, we derive common observations from the various network flexibility proposals, e.g., the importance of adaptation time and the relationship of flexibility with other networking performance indicators. We also relate flexibility to cost, which provides better understanding of the price that we have to pay in order to face changes in the network.

We identify a wide range of future research directions towards flexible communication networks. Among them, we consider that a quantification framework is especially important for the future improvement of network flexibility, as it would enable to measure and compare flexibility among different design choices. Besides, several trade-offs, e.g., overprovision vs. adaptation and software vs. hardware, need to be exploited in order to blueprint flexible network architectures. Applying hot topics such as AI and management automation would further improve the network's ability to accommodate future unknown changes.

Acknowledgment

This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets). The authors alone are responsible for the content of the paper.

References

[1] W. Kellerer, A. Basta *et al.*, "How to measure network flexibility? a proposal for evaluating softwarized networks," *IEEE Communications Magazine*, 2018.

[2] Sdxcentral. Carriers 5G Plans are Rooted in SDN/NFV, Says Ixia Survey. [Online]. Available: https://www.sdxcentral.com/articles/news/carriers-5g-plans-rooted-sdnnfv-says-ixia-survey/2017/09/?c_action=related_articles

[3] BDO International Limited, "2017 telecommunications risk factor survey," Tech. Rep., 2017. [Online]. Available: https://www.bdo.global/en-gb/insights/global-industries/technology,-life-sciences,-media-entertainment-a/2017-telecommunications-risk-factor-survey

[4] D. Kreutz, F. M. Ramos *et al.*, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] R. Mijumbi, J. Serrat *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

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

[7] M. He, A. Basta *et al.*, "How Flexible is Dynamic SDN Control Plane?" in *Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017.

[8] W. Kellerer, A. Basta *et al.*, "Using a flexibility measure for network design space analysis of SDN and NFV," in *Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2016, pp. 423–428.

[9] A. Gamba and A. Triantis, "The value of financial flexibility," *The Journal of Finance*, vol. 63, no. 5, pp. 2263–2296, 2008.

[10] R. D. Shachter and M. Mandelbaum, "A measure of decision flexibility," in *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1996, pp. 485–491.

[11] W. J. Hopp, S. M. Iravani *et al.*, "Vertical flexibility in supply chains," *Management Science*, vol. 56, no. 3, pp. 495–502, 2010.

[12] S. C. Graves and B. T. Tomlin, "Process flexibility in supply chains," *Management Science*, vol. 49, no. 7, pp. 907–919, 2003.

[13] W. Golden and P. Powell, "Towards a definition of flexibility: in search of the holy grail?" *Omega*, vol. 28, no. 4, pp. 373–384, 2000.

[14] S. Peng, L. Shen *et al.*, "User-oriented measurement of software flexibility," in *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering*, vol. 7. IEEE, 2009, pp. 629–633.

[15] H. Subramaniam and H. Zulzalil, "Software quality assessment using flexibility: A systematic literature review," *International Review on Computers and Software*, vol. 7, no. 5, 2012.

[16] A. H. Eden and T. Mens, "Measuring software flexibility," *IEEE Proceedings of Software*, vol. 153, no. 3, pp. 113–125, 2006.

[17] B. A. A. Nunes, M. Mendonca *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[18] W. Xia, Y. Wen *et al.*, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[19] F. Hu, Q. Hao *et al.*, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[20] H. Farhady, H. Lee *et al.*, "Software-defined networking: A survey," *Computer Networks*, vol. 81, pp. 79–95, 2015.

[21] Q. Yan, F. R. Yu *et al.*, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2016.

[22] S. Scott-Hayward, S. Natarajan *et al.*, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[23] I. Ahmad, S. Namal *et al.*, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[24] M. Coughlin, "A survey of SDN security research," *University of Colorado Boulder*, 2014.

[25] M. Chen, Y. Qian *et al.*, "Software-defined mobile networks security," *Mobile Networks and Applications*, vol. 21, no. 5, pp. 729–743, 2016.

[26] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 325–346, 2017.

[27] Y. Jarraya, T. Madi *et al.*, "A survey and a layered taxonomy of software-defined networking," *IEEE Communications surveys & tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[28] A. Lara, A. Kolasani *et al.*, "Network innovation using openflow: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[29] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713–2737, 2016.

[30] S. Costanzo, L. Galluccio *et al.*, "Software defined wireless networks: Unbridling sdns," in *Proceedings of the 2012 European Workshop on Software Defined Networking (EWSDN)*. IEEE, 2012, pp. 1–6.

[31] S. Tomovic, M. Pejanovic-Djurisic *et al.*, "SDN based mobile networks: concepts and benefits," *Wireless Personal Communications*, vol. 78, no. 3, pp. 1629–1644, 2014.

[32] T. Chen, M. Matinmikko *et al.*, "Software defined mobile networks: concept, survey, and research directions," *IEEE Communications Magazine*, vol. 53, no. 11, pp. 126–133, 2015.

[33] V.-G. Nguyen, T.-X. Do *et al.*, "SDN and virtualization-based LTE mobile network architectures: A comprehensive survey," *Wireless Personal Communications*, vol. 86, no. 3, pp. 1401–1438, 2016.

[34] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[35] W. Yang and C. Fung, "A survey on security in network functions virtualization," in *Proceedings of the 2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 15–19.

[36] H. Jang, J. Jeong *et al.*, "A survey on interfaces to network security functions in network virtualization," in *Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2015, pp. 160–163.

[37] V.-G. Nguyen, A. Brunstrom *et al.*, "Sdn/nfv-based mobile packet core network architectures: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1567–1602, 2017.

[38] X. Li and C. Qian, "A survey of network function placement," in *Proceedings of the 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 948–953.

[39] A. Fischer, J. F. Botero *et al.*, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[40] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 358–380, 2015.

[41] M. Yang, Y. Li *et al.*, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 4–18, 2015.

[42] C. Liang, F. R. Yu *et al.*, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE network*, vol. 29, no. 3, pp. 68–74, 2015.

[43] Q. Duan, Y. Yan *et al.*, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Transactions on Network and Service Management*, vol. 9, no. 4, pp. 373–392, 2012.

[44] A. Blenk, A. Basta *et al.*, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685, 2016.

[45] C. Hedrick and L. Bosack, "An introduction to IGRP," *Rutgers-The State University of New Jersey Technical Publication, Laboratory for Computer Science*, 1991.

[46] J. T. Moy, *OSPF: anatomy of an Internet routing protocol*. Addison-Wesley Professional, 1998.

[47] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[48] N. McKeown, T. Anderson *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[49] R. Enns, "Netconf configuration protocol," Internet Requests for Comments, RFC Editor, RFC 4741, December 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4741.txt

[50] B. Pfaff and B. Davie, "The open vswitch database management protocol," Internet Requests for Comments, RFC Editor, RFC 7047, December 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7047.txt

[51] Open network foundation. [Online]. Available: https://www.opennetworking.org/

[52] *OpenFlow Switch Specifications 1.5.1*, Open Networking Foundation, 3 2015.

[53] S. Sharma, D. Staessens *et al.*, "Openflow: Meeting carrier-grade recovery requirements," *Computer Communications*, vol. 36, no. 6, pp. 656–665, 2013.

[54] P. Bosshart, D. Daly *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[55] N. Gude, T. Koponen *et al.*, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[56] Ryu: Component-based software defined networking framework. [Online]. Available: https://osrg.github.io/ryu/

[57] Project floodlight. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[58] S. H. Yeganeh, A. Tootoonchian *et al.*, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[59] S. Sezer, S. Scott-Hayward *et al.*, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[60] T. Koponen, M. Casado *et al.*, "Onix: A distributed control platform for large-scale production networks." in *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, vol. 10, 2010, pp. 1–6.

[61] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, 2010, pp. 3–3.

[62] P. Berde, M. Gerola *et al.*, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the Third ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2014, pp. 1–6.

[63] J. Medved, R. Varga *et al.*, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceedings of the 2014 IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2014, pp. 1–6.

[64] The linux foundation. [Online]. Available: https://www.linuxfoundation.org/

[65] P. P.-S. Chen, "The entity-relationship modeltoward a unified view of data," *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 1, pp. 9–36, 1976.

[66] D. Levin, A. Wundsam *et al.*, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the First ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2012, pp. 1–6.

[67] T. Zhang, A. Bianco *et al.*, "The Role of Inter-Controller Traffic for Placement of Distributed SDN Controllers," *arXiv preprint arXiv:1605.09268*, 2016.

[68] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," *Computer Networks*, vol. 112, pp. 279–293, 2017.

[69] C. Trois, M. D. Del Fabro *et al.*, "A survey on sdn programming languages: toward a taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2687–2712, 2016.

[70] M. F. Bari, S. R. Chowdhury *et al.*, "On orchestrating virtual network functions," in *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 50–56.

[71] ETSI. Network Function Virtualiyation (NFV): Use Cases. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[72] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine*, vol. 47, no. 7, 2009.

[73] A. Khan, A. Zugenmaier *et al.*, "Network virtualization: a hypervisor for the internet?" *IEEE Communications Magazine*, vol. 50, no. 1, 2012.

[74] 3GPP, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions," 3rd Generation Partnership Project (3GPP), TR 36.902, Apr. 2011. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36902.htm

[75] C. Cox, *n Introduction to LTE: LTE, LTE-Advanced, SAE and 4G Mobile Communications*. Wiley, 2012.

[76] M. Koibuchi, A. Funahashi *et al.*, "L-turn routing: An adaptive routing in irregular networks," in *Proceedings of the 2001 IEEE International Conference onParallel Processing*. IEEE, 2001, pp. 383–392.

[77] I. Theiss and O. Lysne, "Froots: a fault tolerant and topology-flexible routing technique," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 10, pp. 1136–1150, 2006.

[78] J. Flich, T. Skeie *et al.*, "A survey and evaluation of topology-agnostic deterministic routing algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 405–425, 2012.

[79] D.-N. Yang and W. Liao, "Optimizing state allocation for multicast communications," in *Proceedings of the 2004 IEEE Conference on Computer Communications (INFOCOM)*, vol. 4. IEEE, 2004, pp. 2719–2730.

[80] ——, "On bandwidth-efficient overlay multicast," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 11, 2007.

[81] F. Pianese, J. Keller *et al.*, "Pulse, a flexible p2p live streaming system," in *Proceedings of the 2006 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2006, pp. 1–6.

[82] A. P. C. da Silva, E. Leonardi *et al.*, "Chunk distribution in mesh-based large-scale p2p streaming systems: A fluid approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 451–463, 2011.

[83] W. Sun, Z. Yang *et al.*, "Hello: A generic flexible protocol for neighbor discovery," in *Proceedings of the 2014 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2014, pp. 540–548.

[84] Y. Zhang, Z. Zhang *et al.*, "Hc-bgp: A light-weight and flexible scheme for securing prefix ownership," in *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 23–32.

[85] K. Xu, H. Liu *et al.*, "One more weight is enough: Toward the optimal traffic engineering with ospf," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2011, pp. 836–846.

[86] A. De Toni and S. Tonchia, "Manufacturing flexibility: a literature review," *International journal of production research*, vol. 36, no. 6, pp. 1587–1617, 1998.

[87] A. K. Sethi and S. P. Sethi, "Flexibility in manufacturing: a survey," *International journal of flexible manufacturing systems*, vol. 2, no. 4, pp. 289–328, 1990.

[88] P. H. Brill and M. Mandelbaum, "On measures of flexibility in manufacturing systems," *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, vol. 27, no. 5, pp. 747–756, 1989.

[89] R. P. Parker and A. Wirth, "Manufacturing flexibility: measures and relationships," *European journal of operational research*, vol. 118, no. 3, pp. 429–449, 1999.

[90] Y. Xiao, "Flexibility measure analysis of supply chain," *International Journal of Production Research*, vol. 53, no. 10, pp. 3161–3174, 2015.

[91] X.-N. Nguyen, D. Saucez *et al.*, "Optimizing rules placement in openflow networks: Trading routing for better efficiency," in *Proceedings of the Third ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2014, pp. 127–132.

[92] Y. Guo, Z. Wang *et al.*, "Traffic engineering in SDN/OSPF hybrid network," in *Proceedings of the 2014 IEEE 22nd International Conference on Network Protocols (ICNP)*. IEEE, 2014, pp. 563–568.

[93] D. Lee, P. Hong *et al.*, "RPA-RA: A resource preference aware routing algorithm in software defined network," in *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[94] K. Toumi, M. S. Idrees *et al.*, "Usage Control Policy Enforcement in SDN-Based Clouds: A Dynamic Availability Service Use Case," in *Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications*. IEEE, 2016, pp. 578–585.

[95] Z. Su, T. Wang *et al.*, "Cheetahflow: Towards low latency software-defined network," in *Proceedings of the 2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 3076–3081.

[96] K.-T. Kuo, C. H.-P. Wen *et al.*, "SWF: Segmented Wildcard Forwarding for flow migration in OpenFlow datacenter networks," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 313–318.

[97] C. Cascone, L. Pollini *et al.*, "Traffic management applications for stateful SDN data plane," in *Proceedings of the 2015 Fourth European Workshop on Software Defined Networks (EWSDN)*. IEEE, 2015, pp. 85–90.

[98] K. He, J. Khalid *et al.*, "Mazu: Taming latency in software defined networks," *University of Wisconsin-Madison Technical Report*, 2014.

[99] A. Mohammadkhan, S. Ghapani *et al.*, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *Proceedings of the 2015 IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2015, pp. 1–6.

[100] W. Zhang, G. Liu *et al.*, "Sdnfv: Flexible and dynamic software defined control of an application-and flow-aware data plane," in *Proceedings of the 17th International Middleware Conference*. ACM, 2016, p. 2.

[101] A. Basta, A. Blenk *et al.*, "Towards a dynamic sdn virtualization layer: A control path migration protocol," in *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 354–359.

[102] K. Sood, S. Yu *et al.*, "Control layer resource management in SDN-IoT networks using multi-objective constraint," in *Proceedings of the 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2016, pp. 71–76.

[103] X. Jin, L. Li *et al.*, "Cellsdn: Software-defined cellular core networks," *Proceedings of the Open Networking Summit SDN Event*, 2013.

[104] X. Jin, L. E. Li *et al.*, "Softcell: Scalable and flexible cellular core network architecture," in *Proceedings of the ninth ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. ACM, 2013, pp. 163–174.

[105] K. Pentikousis, Y. Wang *et al.*, "Mobileflow: Toward software-defined mobile networks," *IEEE Communications magazine*, vol. 51, no. 7, pp. 44–53, 2013.

[106] I. F. Akyildiz, P. Wang *et al.*, "Softair: A software defined networking architecture for 5g wireless systems," *Computer Networks*, vol. 85, pp. 1–18, 2015.

[107] Z. A. Qazi, P. K. Penumarthi *et al.*, "Klein: A minimally disruptive design for an elastic cellular core," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 2.

[108] H. Wang, S. Chen *et al.*, "Softnet: A software defined decentralized mobile network architecture toward 5g," *IEEE Network*, vol. 29, no. 2, pp. 16–22, 2015.

[109] M. Gramaglia, I. Digon *et al.*, "Flexible connectivity and QoE/QoS management for 5G Networks: The 5G NORMA view," in *Proceedings of the 2016 IEEE International Conference on Communications Workshop*. IEEE, 2016, pp. 373–379.

[110] J. Hwang, K. K. Ramakrishnan *et al.*, "NetVM: high performance and flexible networking using virtualization on commodity platforms," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 34–47, 2015.

[111] K. Sundaresan, M. Y. Arslan *et al.*, "FluidNet: A flexible cloud-based radio access network for small cells," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 915–928, 2016.

[112] X. Foukas, N. Nikaein *et al.*, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016, pp. 427–441.

[113] A. Checko, H. L. Christiansen *et al.*, "Cloud RAN for mobile networksA technology overview," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 405–426, 2015.

[114] V. N. Ha, L. B. Le *et al.*, "Coordinated multipoint transmission design for cloud-RANs with limited fronthaul capacity constraints," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7432–7447, 2016.

[115] T. X. Tran and D. Pompili, "Dynamic Radio Cooperation for User-Centric Cloud-RAN With Computing Resource Sharing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2379–2393, 2017.

[116] J. Tang, W. P. Tay *et al.*, "Cross-layer resource allocation with elastic service scaling in cloud radio access network," *IEEE Transactions on Wireless Communications*, vol. 14, no. 9, pp. 5068–5081, 2015.

[117] K. Wang and Y. Cen, "Real-Time Partitioned Scheduling in Cloud-RAN with Hard Deadline Constraint," in *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2017, pp. 1–6.

[118] S. Bhatia, M. Motiwala *et al.*, "Trellis: A platform for building flexible, fast virtual networks on commodity hardware," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 72.

[119] H. Rodrigues, J. R. Santos *et al.*, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks." in *Proceedings of the USENIX Third Workshop on IO Virtualization (WIOV)*, 2011.

[120] R. Sherwood, G. Gibb *et al.*, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, vol. 1, p. 132, 2009.

[121] X. Jin, J. Gossels *et al.*, "Covisor: A compositional hypervisor for software-defined networks." in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2015.

[122] W. Shen, M. Yoshida *et al.*, "vConductor: An NFV management solution for realizing end-to-end virtual network services," in *Proceedings of the 2014 16th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2014, pp. 1–6.

[123] S. Clayman, E. Maini *et al.*, "The dynamic placement of virtual network functions," in *Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–9.

[124] Y. Chang, A. Rezaei *et al.*, "Hydra: Leveraging Functional Slicing for Efficient Distributed SDN Controllers," *arXiv preprint arXiv:1609.07192*, 2016.

[125] V. Riccobene, A. Lombardo *et al.*, "Network functions at the edge (NetFATE): design and implementation issues," *National Telecommunications and Information Theory Group (GTTI)*, 2014.

[126] M. Xia, M. Shirazipour *et al.*, "Network function placement for NFV chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[127] A. Basta, W. Kellerer *et al.*, "A virtual SDN-enabled LTE EPC architecture: A case study for S-/P-gateways functions," in *Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. IEEE, 2013, pp. 1–7.

[128] ——, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proceedings of the 4th Workshop on All things cellular: Operations, Applications, & Challenges*. ACM, 2014, pp. 33–38.

[129] T. Taleb, M. Bagaa *et al.*, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 3879–3884.

[130] M. Bagaa, T. Taleb *et al.*, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Proceedings of the 2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014, pp. 2402–2407.

[131] A. Baumgartner, V. S. Reddy *et al.*, "Combined virtual mobile core network function placement and topology optimization with latency bounds," in *Proceedings of the 2015 Fourth European Workshop on Software Defined Networks (EWSDN)*. IEEE, 2015, pp. 97–102.

[132] D. Sabella, P. Rost *et al.*, "RAN as a service: Challenges of designing a flexible RAN architecture in a cloud-based heterogeneous mobile network," in *Proceedings of the 2013 Future Network and Mobile Summit*. IEEE, 2013, pp. 1–8.

[133] A. Maeder, M. Lalam *et al.*, "Towards a flexible functional split for cloud-RAN networks," in *Proceedings of the 2014 European Conference on Networks and Communications (EuCNC)*. IEEE, 2014, pp. 1–5.

[134] G. Mountaser, M. L. Rosas *et al.*, "On the feasibility of MAC and PHY split in Cloud RAN," in *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2017, pp. 1–6.

[135] C.-Y. Chang, N. Nikaein *et al.*, "FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.

[136] W.-L. Yeow, C. Westphal *et al.*, "Designing and embedding reliable virtual infrastructures," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 57–64, 2011.

[137] A. Leivadeas, C. Papagianni *et al.*, "Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1077–1086, 2013.

[138] M. Yu, Y. Yi *et al.*, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[139] N. M. K. Chowdhury, M. R. Rahman *et al.*, "Virtual network embedding with coordinated node and link mapping," in *Proceedings of the 2009 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2009, pp. 783–791.

[140] A. Ludwig, S. Schmid *et al.*, "Specificity vs. flexibility: On the embedding cost of a virtual network," in *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*. IEEE, 2013, pp. 1–9.

[141] G. Yao, J. Bi *et al.*, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.

[142] M. T. I. ul Huque, W. Si *et al.*, "Large-scale dynamic controller placement," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 63–76, 2017.

[143] A. Dixit, F. Hao *et al.*, "ElastiCon; an elastic distributed SDN controller," in *Proceedings of the 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2014, pp. 17–27.

[144] A. Gudipati, D. Perry *et al.*, "SoftRAN: Software defined radio access network," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 25–30.

[145] G. A. Carella, M. Pauls *et al.*, "An extensible Autoscaling Engine (AE) for Software-based Network Functions," in *Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2016, pp. 219–225.

[146] M. Yoshida, W. Shen *et al.*, "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure," in *Proceedings of the 2014 16th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2014, pp. 1–6.

[147] V. Shrivastava, P. Zerfos *et al.*, "Application-aware virtual machine migration in data centers," in *Proceedings of the 2011 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2011, pp. 66–70.

[148] D. Williams, H. Jamjoom *et al.*, "The Xen-Blanket: virtualize once, run everywhere," in *Proceedings of the 7th ACM European Conference on Computer Systems*. ACM, 2012, pp. 113–126.

[149] M. Qian, W. Hardjawana *et al.*, "Baseband processing units virtualization for cloud radio access networks," *IEEE Wireless Communications Letters*, vol. 4, no. 2, pp. 189–192, 2015.

[150] N. Zhang, N. Cheng *et al.*, "Cloud assisted HetNets toward 5G wireless networks," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 59–65, 2015.

[151] M. Chen, Y. Zhang *et al.*, "Cloud-based wireless network: Virtualized, reconfigurable, smart wireless network to enable 5G technologies," *Mobile Networks and Applications*, vol. 20, no. 6, pp. 704–712, 2015.

[152] Y. Fengyi, W. Haining *et al.*, "A flexible three clouds 5G mobile network architecture based on NFV & SDN," *China Communications*, vol. 12, no. Supplement, pp. 121–131, 2015.

[153] I. Fajjari, N. Aitsaadi *et al.*, "Adaptive-VNE: A flexible resource allocation for virtual network embedding algorithm," in *Proceedings of the 2012 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2012, pp. 2640–2646.

[154] J. He, R. Zhang-Shen *et al.*, "Davinci: Dynamically adaptive virtual networks for a customized internet," in *Proceedings of the 2008 ACM CONEXT Conference*. ACM, 2008, p. 15.

[155] Y. Wei, J. Wang *et al.*, "Bandwidth allocation in virtual network based on traffic prediction," in *Proceedings of the 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. IEEE, 2010, pp. 1–4.

[156] A. Blenk, A. Basta *et al.*, "HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation," in *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 397–405.

[157] M. P. Grosvenor, M. Schwarzkopf *et al.*, "Queues Don'T Matter when You Can JUMP Them!" in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, ser. NSDI'15. Berkeley, CA, USA: USENIX Association, 2015, pp. 1–14.

[158] A. Ksentini and N. Nikaein, "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 102–108, 2017.

[159] J. He and W. Song, "AppRAN: Application-oriented radio access network sharing in mobile networks," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 3788–3794.

[160] T. Guo and R. Arnott, "Active LTE RAN sharing with partial resource reservation," in *Proceedings of the 2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*. IEEE, 2013, pp. 1–5.

[161] I. Fajjari, N. Aitsaadi *et al.*, "VNR algorithm: A greedy approach for virtual networks reconfigurations," in *Proceedings of the 2011 IEEE*

Global Telecommunications Conference (GLOBECOM). IEEE, 2011, pp. 1–6.

[162] N. F. Butt, M. Chowdhury *et al.*, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *Proceedings of the International Conference on Research in Networking*. Springer, 2010, pp. 27–39.

[163] N. Shahriar, R. Ahmed *et al.*, "Generalized recovery from node failure in virtual network embedding," *IEEE Transactions on Network and Service Management*, 2017.

[164] E. Keller, S. Ghorbani *et al.*, "Live migration of an entire network (and its hosts)," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets)*. ACM, 2012, pp. 109–114.

[165] S. Ghorbani, C. Schlesinger *et al.*, "Transparent, live migration of a software-defined network," in *Proceedings of the 2014 ACM Symposium on Cloud Computing*. ACM, 2014, pp. 1–14.

[166] G. Bianchi, M. Bonola *et al.*, "Openstate: programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 44–51, 2014.

[167] H. Mao, M. Alizadeh *et al.*, "Resource management with deep reinforcement learning." in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets)*, 2016, pp. 50–56.

[168] A. Basta, A. Blenk *et al.*, "Towards a cost optimal design for a 5G mobile core network based on SDN and NFV," *IEEE Transactions on Network and Service Management*, 2017.

[169] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components." in *Proceedings of the 2006 IEEE Conference on Computer Communications (INFOCOM)*, no. 2006. IEEE, 2006, pp. 1–12.

[170] M. Rost, S. Schmid *et al.*, "It's about time: On optimal virtual network embeddings under temporal flexibilities," in *The Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 17–26.

[171] (2017) Open Baton - An Extensible and Customizable NFV MANO-Compliant Framework. [Online]. Available: https://openbaton.github.io/

[172] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *Proceedings of the 2006 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2006, pp. 1–12.

[173] J. Inführ and G. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints," *Network optimization*, pp. 105–117, 2011.

[174] C. C. Marquezan, L. Z. Granville *et al.*, "Distributed autonomic resource management for network virtualization," in *Proceedings of the 2010 IEEE Network operations and management symposium (NOMS)*. IEEE, 2010, pp. 463–470.

[175] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*. Pearson Education, 2012.

[176] X. Meng, V. Pappas *et al.*, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the 2010 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2010, pp. 1–9.

[177] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[178] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. 100, no. 10, pp. 892–901, 1985.

[179] C. Guo, G. Lu *et al.*, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[180] A. Singla, C.-Y. Hong *et al.*, "Jellyfish: Networking data centers, randomly." in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 12, 2012, pp. 17–17.

[181] I. Fajjari, N. A. Saadi *et al.*, "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic," in *Proceedings of the 2011 IEEE International Conference on Communications (ICC)*. IEEE, 2011, pp. 1–6.

[182] R. Kokku, R. Mahindra *et al.*, "CellSlice: Cellular wireless resource slicing for active RAN sharing," in *Proceedings of the 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2013, pp. 1–10.

[183] A. Khan, W. Kellerer *et al.*, "Network sharing in the next mobile network: TCO reduction, management flexibility, and operational independence," *IEEE Communications Magazine*, vol. 49, no. 10, 2011.

[184] E. Hernandez-Valencia, S. Izzo *et al.*, "How will NFV/SDN transform service provider opex?" *IEEE Network*, vol. 29, no. 3, pp. 60–67, 2015.

[185] P. Ruth, J. Rhee *et al.*, "Autonomic live adaptation of virtual networked environments in a multidomain infrastructure," *Journal of Internet Services and Applications*, vol. 2, no. 2, pp. 141–154, 2011.

[186] O. Hanseth, E. Monteiro *et al.*, "Developing information infrastructure: The tension between standardization and flexibility," *Science, Technology, & Human Values*, vol. 21, no. 4, pp. 407–426, 1996.

[187] N. Huber, M. von Quast *et al.*, "Evaluating and modeling virtualization performance overhead for cloud environments." in *Proceedings of the International Conference on Cloud Computing and Services Science (CLOSER)*, 2011, pp. 563–573.

[188] M. Casado, M. J. Freedman *et al.*, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[189] J.-L. Chen, Y.-W. Ma *et al.*, "Software-defined network virtualization platform for enterprise network resource management," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 179–186, 2016.

[190] X. Zhou, R. Li *et al.*, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.

[191] M. H. Rehmani, M. E. Kantarci *et al.*, "Ieee access special section editorial smart grids: A hub of interdisciplinary research," *IEEE Access*, vol. 3, pp. 3114–3118, 2015.

[192] P. K. Agyapong, M. Iwamura *et al.*, "Design considerations for a 5G network architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, 2014.

[193] K. Samdanis, X. Costa-Perez *et al.*, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[194] H. Ji, S. Park *et al.*, "Ultra-reliable and low-latency communications in 5g downlink: Physical layer aspects," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 124–130, 2018.

[195] J. Ordonez-Lucena, P. Ameigeiras *et al.*, "Network slicing for 5g with sdn/nfv: concepts, architectures and challenges," *arXiv preprint arXiv:1703.04676*, 2017.

[196] H. I. Kobo, A. M. Abu-Mahfouz *et al.*, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[197] L. Galluccio, S. Milardo *et al.*, "Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 513–521.

[198] N. Bizanis and F. A. Kuipers, "Sdn and virtualization solutions for the internet of things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[199] J. Liu, Y. Li *et al.*, "Software-defined internet of things for smart urban sensing," *IEEE communications magazine*, vol. 53, no. 9, pp. 55–63, 2015.

[200] N. Omnes, M. Bouillon *et al.*, "A programmable and virtualized network & IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges," in *Proceedings of the 2015 18th International Conference on Intelligence in Next Generation Networks (ICIN)*. IEEE, 2015, pp. 64–69.

[201] Y. Koh, R. Knauerhase *et al.*, "An analysis of performance interference effects in virtual environments," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*. IEEE, 2007, pp. 200–209.

[202] A. Tchana, B. Bui *et al.*, "Mitigating performance unpredictability in the IaaS using the Kyoto principle," in *Proceedings of the 17th International Middleware Conference*. ACM, 2016, p. 6.

[203] Pica8. (2017) Whitepaper: Bare Metal NetworkingLeveraging White BoxThinking. [Online]. Available: http://www.pica8.com/documents/pica-whitepaper-white-box.pdf

[204] DELL and Cumulus. (2017) Whitepaper: The Economic Advantages of Open and Web-Scale Networking. [Online]. Available: http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell-EMC-Cumulus-Open-Networking-White-Paper.pdf

[205] M. He, A. Basta *et al.*, "P4nfv: An nfv architecture with flexible data plane reconfiguration," in *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 90–98.

[206] I. Jang, D. Suh *et al.*, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE Journal on*

Selected Areas in Communications, vol. 35, no. 11, pp. 2532–2541, 2017.

[207] A. Blenk, P. Kalmbach *et al.*, "Boost online virtual network embedding: Using neural networks for admission control," in *The Proceedings of the 2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 2016, pp. 10–18.

[208] J. Zerwas, P. Kalmbach *et al.*, "Ahab: Data-driven virtual cluster hunting," in *Proceedings of the 17th IFIP Networking Conference*, 2018.

[209] A. Blenk, P. Kalmbach *et al.*, "o'zapft is: Tap your network algorithm's big data!" in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. ACM, 2017, pp. 19–24.

[210] M. He, P. Kalmbach *et al.*, "Algorithm-data driven optimization of adaptive communication networks," in *Proceedings of the 2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 2017, pp. 1–6.

[211] L. Cao, P. Sharma *et al.*, "NFV-vital: A framework for characterizing the performance of virtual network functions," in *Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 93–99.

TABLE VII: TABLE OF ALL PROPOSALS.

Prop.	Tech.	Cat.	Asp.	Dom.	Pl.	Main objective	Adaptation Cost	Adp. Time
[91]	SDN	Config	FlC	WAN	DP	Accommodate high traffic rate	Computation, extra forwarding latency	-
[92]	SDN	Config	FlC	WAN	DP	Link load balancing	-	-
[93]	SDN	Config	FlC	WAN	DP	Link load balancing	-	-
[94]	SDN	Config	FlC	WAN	DP	Policy enforcement	Overhead of flow rules managing	✓
[95]	SDN	Config	FlC	WAN	DP	Forwarding latency improvement	Computation, overhead of flow rules managing	-
[96]	SDN	Config	FlC	DC	DP	Forwarding latency improvement	Overhead of flow rules managing	-
[97]	SDN	Config	FlC	WAN	DP	Forwarding latency improvement	Extra inter-switch signaling	-
[98]	SDN	Config	FlC	WAN	DP	Failure recovery	-	✓
[99]	SDN	Config, Locat	FlC, FuP	WAN	DP	NFV management and routing	-	-
[100]	SDN	Config	FlC	WAN	DP	General routing	Extra latency of flow redirection	✓
[101]	SDN	Config	FlC	WAN	CP	Control plane management	Sync overhead, extra latency of flow migration	✓
[102]	SDN	Config	FlC	AN	CP	Control plane management	-	-
[103]	SDN	Config	FlC, FuC	MCN	DP, CP	Flow aggregation	Additional equipment and steering strategy	-
[104]	SDN	Config	FlC, FuC	MCN	CP	Traffic and upgrades adaptation	Additional equipment and steering strategy	-
[105]	SDN	Config	FlC	MCN	DP, CP	Traffic adaptation	Additional equipment and steering strategy	-
[106]	SDN	Config	FlC, RFS	MCN, RAN	DP, CP	Traffic adaptation	-	-
[107]	SDN	Config	FlC, FuP	MCN	DP, CP	Core load balancing	Computation time, extra delays, impact on standards	✓
[108]	SDN	Config	FlC, FuC	MCN, RAN	DP, CP	User mobility adaptation	Extra delays, additional bandwidth usage, impact on standards	-
[109]	SDN	Config	FlC, FuP	MCN	DP, CP	Traffic adaptation	Additional equipment, suboptimal handovers	-
[110]	NFV	Config	FuC	WAN	DP	Fast packet processing	-	-
[111]	NFV	Config	FuC	RAN	DP, CP	User mobility adaptation	Suboptimal mobility management, additional energy consumption	✓
[112]	NFV	Config, Locat	FuC, PaC, FuP	RAN	CP	Traffic and upgrades adaptation	Signaling overhead, extra delays, security risks	-
[114]	NFV	Config	PaC, RFS	RAN	DP, CP	Transmission power selection	Computation time	-
[115]	NFV	Config	PaC, RFS	RAN	DP, CP	Optimal user clustering	Computation time, better channel estimation and synchronization	✓
[116]	NFV	Config	PaC, RFS	RAN	DP, CP	Optimal allocation of resources	Signaling overhead, high bandwidth links	-
[117]	NFV	Config	PaC, RFS	RAN	DP, CP	Optimal scheduling	Computation time, high CPU usage	-
[118]	NV	Config	PaC	DC	CP	Virtual IP network customization	-	-
[119]	NV	Config	PaC	DC	CP	Performance isolation of VN	Extra configuration latency	-
[120]	NV	Config	PaC	DC	CP	High performance virtual SDN	-	-
[121]	NV	Config	PaC	WAN	CP	Hypervisor design	Computation, overhead of flow rules managing	✓
[122]	NFV	Locat	FuP	WAN	DP	Automatic NFV management	-	-
[123]	NFV	Locat	FuP	WAN	DP	Automatic NFV management	-	-
[124]	NFV	Locat	FuP	WAN	DP	NFV placement	Sync overhead, overhead of flow rules managing	✓
[125]	NFV	Locat	FuP	WAN	DP	NFV placement	Overhead of function, flow migration	-
[126]	NFV	Locat	FuP	DC	DP	NFV placement	-	-
[127]	NFV	Locat	FuP	MCN	CP, DP	Mobile core function placement	-	✓
[128]	NFV	Locat	FuP	MCN	CP, DP	Mobile core function placement	-	-
[129]	NFV	Locat	FuP	MCN	DP, CP	Mobile core function placement	Computation time, additional bandwidth usage	-

Continuation of Table VII

Prop.	Tech.	Cat.	Asp.	Dom.	Pl.	Main objective	Adaptation Cost	Adp. Time
[130]	NFV	Locat	FuP	MCN	DP, CP	Mobile core function placement	Computation time, additional bandwidth usage	-
[131]	NFV	Locat	FuP	MCN	DP, CP	Mobile core function placement	Computation time, additional bandwidth usage	✓
[132]	NFV	Locat	FuP	RAN	DP, CP	RAN function splitting	Low latency links, extra delays, additional equipment	-
[133]	NFV	Locat	FuP	RAN	DP, CP	RAN function splitting	Low latency links, extra delays, additional equipment	-
[134]	NFV	Locat	FuP	RAN	DP, CP	RAN function splitting	Low latency links; extra delays	✓
[135]	NFV	Locat	FuP	RAN	DP, CP	RAN function splitting	Low latency links, extra delays, additional equipment	-
[139]	NV	Locat	FuP	DC	DP	VNE with practical constraints	-	-
[140]	NV	Locat	FuP	DC	DP	VNE with partially specified constraints	-	-
[136]	NV	Locat	FuP	DC	DP	Reliable virtual networks implementation	-	-
[137]	NV	Locat	FuP	DC	DP	VNE with heterogeneous resources	-	-
[138]	NV	Locat, Scale	FuP, ToA	DC	DP	General VNE	Service interruption during migration	✓
[141]	SDN	Scale	RFS	WAN	CP	Controller placement optimization	-	-
[142]	SDN	Scale	RFS	WAN	CP	Controller placement optimization	-	-
[143]	SDN	Scale	RFS	WAN	CP	Controller pool scaling	Packet loss during migration	-
[144]	SDN	Scale	RFS	RAN	CP	Optimal scheduling	Low latency links, extra delays, additional equipment	-
[145]	NFV	Scale	RFS	WAN	DP	Automatic NFV resource scaling	-	✓
[146]	NFV	Scale	RFS	DC	DP	NFV resource management in DC	-	-
[147]	NFV	Scale	RFS	DC	DP	VM management in DC	-	-
[148]	NFV	Scale	RFS	DC	DP	Data center virtualization	-	✓
[113]	NFV	Scale	RFS	RAN	DP, CP	RAN functions virtualization	Low latency links, extra delays	-
[149]	NFV	Scale	RFS	RAN	DP, CP	RAN functions virtualization	Computation time	-
[150]	NFV	Scale	RFS	RAN	DP, CP	RAN functions virtualization	Security risks, suboptimal mobility management	-
[151]	NFV	Scale	RFS	RAN	DP, CP	RAN functions virtualization	-	-
[152]	NFV	Scale	RFS	RAN	DP, CP	RAN functions virtualization	-	-
[153]	NV	Scale	RFS	WAN	DP	Adaptive bandwidth reservation	-	-
[154]	NV	Scale	RFS	WAN	DP	Dynamic link bandwidth adaptation	-	✓
[155]	NV	Scale	RFS	WAN	DP	Adaptive bandwidth reservation	-	-
[156]	NV	Scale	RFS	WAN	CP	Virtual SDN hypervisor	-	-
[157]	NV	Scale	RFS	DC	DP	Virtual network provision	-	-
[158]	NV	Scale	RFS	RAN	DP, CP	RAN slicing	-	-
[159]	NV	Scale	RFS	RAN	DP, CP	RAN slicing	Computation time	-
[160]	NV	Scale	RFS	RAN	DP, CP	RAN slicing	Suboptimal network utilization	-
[161]	NV	Scale	ToA	WAN	DP	Heuristic VNE algorithm	Service interruption during migration	-
[162]	NV	Scale	ToA	WAN	DP	Re-optimize VNE.	-	-
[163]	NV	Scale	ToA	WAN	DP	VN failure recovery	Computation	-
[164]	NV	Scale	ToA	DC	DP	VM migration mechanism	Service interruption during migration	-
[165]	NV	Scale	ToA	DC	CP, DP	Virtual SDN migration mechanism	Packet loss during vSDN migration	-