How to measure network flexibility? - A proposal for evaluating softwarized networks

Wolfgang Kellerer, Arsany Basta, Péter Babarczi, Andreas Blenk, Mu He, Markus Klügel, Alberto Martínez Alba
Chair of Communication Networks, Department of Electrical and Computer Engineering,
Technical University of Munich, Germany
{wolfgang.kellerer, arsany.basta, peter.babarczi, andreas.blenk, mu.he,
markus.kluegel, alberto.martinez-alba}@tum.de

Abstract

The emerging trend to softwarize networks based on concepts such as Network Virtualization, Software Defined Networking and Network Functions Virtualization promises to increase flexibility in networking. So far, flexibility is used rather as a qualitative argument for a network design choice. Furthermore, the meaning of flexibility behind such qualitative arguments is highly varying in the state of the art as a common understanding of flexibility is missing. In this paper, we present an approach towards evaluating network flexibility through a definition of a flexibility measure, which provides a quantitative analysis and a comparison of different network designs. For us, network flexibility refers to the ability to support new requests that can be, for example, changes in the requirements or new traffic distributions. We show with two case studies how an application of such measure could lead to a better understanding of different network designs with respect to flexibility. We also illustrate the trade-off between flexibility and cost needed to provide flexibility. With our proposed flexibility measure, we would like to stimulate the discussion towards a more quantitative analysis of softwarized networks and beyond.

I. Introduction

In recent years, network operation has become more software-oriented. Concepts such as Network Virtualization (NV), Software Defined Networking (SDN), and Network Functions Virtualization (NFV) provide a new level of indirection and new interfaces for setting up (virtual) networks and (virtual) network functions on demand. These concepts support the adaptation of networks to changes that arise from user demands as well as from traffic fluctuations. When designing networks for adaptation to changes, *flexibility* is claimed to be the competitive advantage of new network designs.

Although researchers claim flexibility, a common comparison of flexibility is not performed due to a missing consensus of how flexibility can be quantified. Thus the available qualitative arguments vary a lot in literature, since the use of flexibility as a measure is not well understood or has not been deeply investigated. A missing definition leaves readers to draw conclusions on their own.

We would like to close this gap by proposing an initial definition of a *flexibility measure* and show its practical application with two case studies.

Such flexibility measure enables a *quantitative* analysis and comparison of different network design choices. Thus, it supports decision making for designs considering softwarization concepts where flexibility is an important design goal due to the increase of design choices in addition to performance metrics, e.g., high throughput or low latency. Today, the rather qualitative statements on flexibility can not fully support decision makers in particular from the industry. For research and development, a flexibility definition fosters the understanding of which technical concepts lead to more flexibility in network design. In this way, design guidelines for flexibility can be established, which we illustrate along with the case studies.

Flexibility commonly refers to the timely support of changes in the requirements. For example, new user demands cause a traffic increase in an enterprise network. One would consider the enterprise network design as flexible if this traffic increase can be accommodated within a short time frame.

For the proposed *flexibility measure*, we refer to changes in the requirements as "*new requests*", which the system was originally not designed for. New requests arise due to designers' demand, such as shorter latency budgets or varying traffic characteristics. To support new requests, the system can adapt the available network resources if needed. Our proposed flexibility measure is based on the ratio of new requests that can be supported within a time frame from the total number of given new requests. This flexibility measure supports a quantitative analysis and comparison of different network designs under the same given set of requests.

Attempting to measure flexibility cannot neglect the *costs* that might be involved making a network design more flexible. Hence, flexibility shall always be evaluated in light of the costs involved. For example, providing several data centers in a network setup is likely to be more flexible with respect to changing demands for network functions, as functions can be migrated to potentially more places. However, installing more data centers may also involve higher cost due to necessary equipment purchases.

The contribution of our work is twofold: (1) We motivate and propose a new measure for a quantitative evaluation of networks with respect to flexibility (Sections II and III) and (2) we apply our proposed measure to two case studies (Section IV).

II. Flexibility of Softwarized Networks in Literature

In this section, we extract examples for the definition of flexibility applied on SDN, NFV and NV and, if described, show how flexibility is expressed within each context and its relation to cost.

SDN has been developed to target programmable flows and to centralize network control, which contributes to flexibility in terms of flow steering and configuration. We can assess such flexibility needs in terms of potential vs. actually possible flow configurations. In fact, OpenFlow (OF) [1], the most commonly used protocol to implement SDN, has an upper bound on its flexibility due to the limited set of configurations specified in each protocol version. For instance, flow configuration based on application layer information, e.g., HTTP requests, from flow packets are not yet available with the latest OF protocol version 1.5 [2]. Another notion of flexibility in SDN is control plane distribution. The framework "Kandoo" [3] for example looks into preserving scalability for SDN networks in data centers. It has two layers of controllers, a lower layer with no knowledge of the network-wide state and an upper layer with a logically centralized controller maintaining network-wide state. Evaluations show that bandwidth and control load savings start from a fanout of four, i.e., one core switch and four top-of-rack switches. Hence, savings promised by the provided flexibility vary across design choices.

With NFV, network functions get developed as software and are executed on commodity hardware. This offers flexibility to define and program network functions more easily [4]. NFV can also provide flexibility in terms of resource scaling, e.g., scale up resources assigned to a network function or scale out a function on multiple hardware entities. Software functions also contribute to the flexibility in the function placement. However, similar to SDN, flexibility of NFV is not absolute and might have a cost impact on performance. [5] shows a trade-off between flexibility and cost of NFV. It investigates virtualization of network middleboxes as software functions running in a cloud. Middleboxes contribute to a large fraction of network domains, thus software inter-changeable middleboxes can promise a huge increase in flexibility. The evaluation shows that flexibility of software middleboxes can induce cost in terms of increased latency depending on the cloud provider and solution taken, up to an additional 52% in the worst case.

NV abstracts network resources from physical infrastructure with the scope of adding flexibility to network resources provisioning. With existing networking hardware, virtualization provides the ability to add or adapt virtual networks when needed, which is argued to provide the needed flexibility to the ossified Internet architecture [6]. Addressing migration, for instance, to evaluate the flexibility of virtual networks, [7] shows a study for live migration of virtual switches. Live migration provides flexibility in adapting the virtual network topology. The evaluation shows that the introduced solution can successfully achieve migration without packet loss, i.e., transparent to the service. However, an extra software layer is added that comes at a control overhead of up to 7%. This means that gains in terms of topology flexibility offered by NV might have drawbacks on the network performance.

Overall, we can observe that flexibility is highly present in literature and, in particular, used as an argument in softwarized networks. However, a common measure for a quantitative analysis of flexibility is missing. Such analysis should be part of a comprehensive analysis of the network design space to show why a design choice is flexible and to what extent.

III. Flexibility as a New Measure

In our proposed flexibility analysis, we compare the flexibility of different network design choices of a system. Hence, flexibility analysis is always specific to the particular *system* under comparison. A system in this context is based on an architecture and a network topology. It implements network algorithms and protocols and it comes with certain resource characteristics such as link capacities and processing capabilities. Running a system comes at a certain cost to which we refer in Section III-C. To analyze network flexibility, we challenge the system with "new requests", i.e., request new requirements or input new traffic distributions and see if they can be supported.

A. Proposal for a Flexibility Measure

We define the flexibility φ of a system S, dependent on its current state, as the fraction of new requests that can be supported from a given set and sequence of new requests, within a given time threshold T. This notion is formalized in Eq.1. As the number of supported requests is always smaller or equal to the number of total requests, the flexibility is always in the interval $[0, 1]$ and can be expressed as a percentage.

$$\varphi_T(S) = \frac{|\text{supported new requests within time threshold } T|}{|\text{total given new requests}|} \tag{1}$$

When the time threshold T becomes infinite, we arrive at a simplified flexibility measure $\varphi_\infty(S)$, which is independent of time. This helps us to assess the ability of a system to support requests in principle. See [8] for an example.

Fig. 1 illustrates a simple example. It shows an SDN network with one controller (CTR). In SDN, the controller is a network function that is deployed on a physical node. The initial controller capacity (cc), i.e., CPU resources required to run the controller, equals to 1. An increase of the control traffic generates a new request to increase the controller capacity to 2 (upper left side figure). This request requires the controller to migrate to a node that has a node capacity (nc), i.e., available

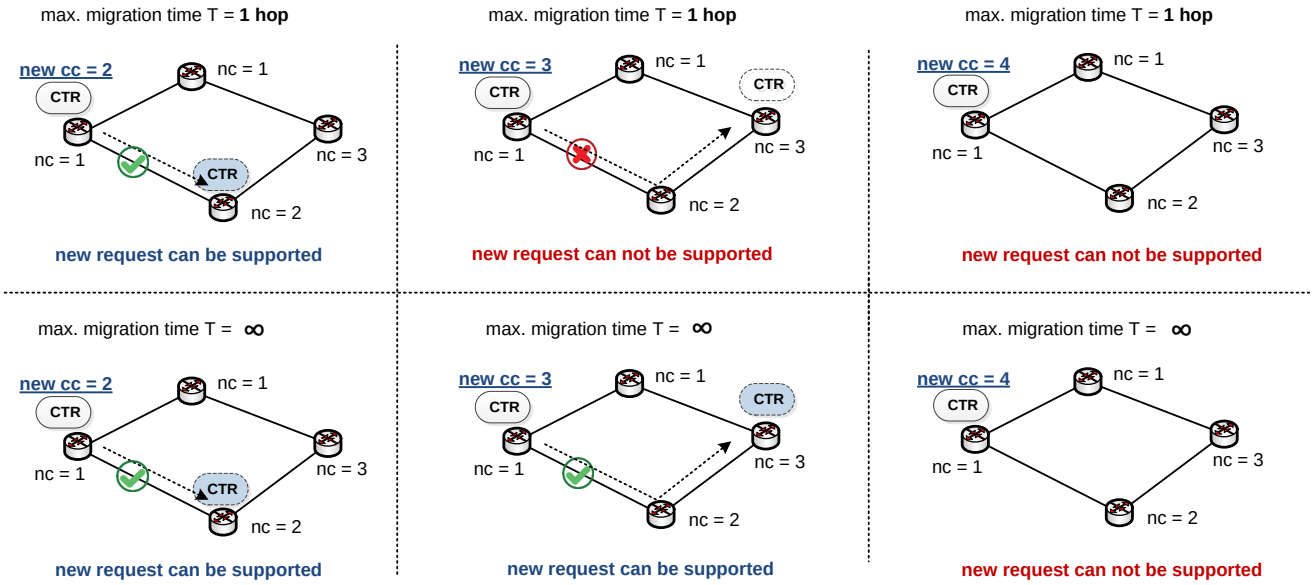


Fig. 1: Flexibility measure example; requests support within a given time threshold T. (**CTR**): SDN controller. (**cc**): controller required capacity, (**nc**): node available capacity.

CPU resources, of 2. In the following, we assume that a migration over a 1 hop link takes 1 time unit. When a maximum migration time threshold T of 1 unit is allowed, the controller would be able to migrate and hence, this network is flexible enough to support the request. In case the new controller capacity request is 3 and the maximum migration time T threshold is 1, the network would fail to support this request, which is shown in the upper middle figure. Even though the network contains a node with a capacity of 3, the system is not flexible enough to support the controller migration within 1 time unit. In case the migration time T is not a constraint, i.e., goes to ∞ (lower middle figure), this network design would be able to support the new controller capacity request of 3. Note that if the request for the controller capacity $= 4$ (both right side figures), this network would not be able to support it regardless of the time T. Hence, with this given set of new requests for controller capacity $(r_1, r_2, r_3) = (2, 3, 4)$, the flexibility of this network under a migration time T of 1 unit is $\varphi_1(S) = 1/3 = 0.33$ (upper figures) and without a requirement on the migration time would be $\varphi_\infty(S) = 2/3 = 0.66$ (lower figures). For a simple illustration, in this example, we consider only a small number of requests that have equal probability of their occurrence. In reality, the set of given requests and their probability distribution may be larger and more complex.

TABLE I: Technical concepts and their support of flexibility in networks. (•: main target).

Category (see Sec. III-B)	Aspect (see Sec. III-B)	SDN	NFV	NV
Adapt Configuration	Flow Config.: flow steering	•		
	Function Config.: function programming		•	
	Parameter Config.: 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			•

B. Flexibility Aspects

Flexibility as a measure is used with a specific objective in mind. To reflect such objectives, we come up with *flexibility aspects* derived from the technologies SDN, NFV and NV (Sec. II). A flexibility aspect describes a concrete ability in which a network can adapt, e.g., change the flow routes or change of the allocated resources. For an insightful flexibility analysis, flexibility aspects shall be as independent to each other as possible.

Table I summarizes the state of the art analysis from Section II. We can observe that network flexibility can be assessed in three categories: flexibility to adapt the configuration, to locate functions and to scale resources.

Configuration can either relate to a single parameter or a state change, i.e., ”re-configuration”, or an addition to the possible set of configurations that the network supports. Network configuration flexibility can be expressed for the configuration of flows, functions and parameters. It can be assessed in terms of the number or set of possible configurations, where a larger set adds to flexibility. Another category is the ability to change the deployment location of network functions, i.e., network

function placement. It enables the support of different latency requirements and the combination of functions, i.e., chaining. A third category is scale. It includes the ability of a network to scale up or down its resources, e.g., to add link capacity, or to scale the allocated resources to network flows, e.g., allocate more capacity, or to scale out network functions, e.g., to increase the number of deployed functions. It also refers to the ability to scale the network topology, e.g., increase the network size through adding nodes and links or change the network connectivity from a tree to a mesh topology.

We initially define the following flexibility aspects within the above categories focusing on SDN, NFV and NV (Table I).

- **Flow Configuration:** adaptation of the course of flows, i.e., steering the flows inside a network through configuring forwarding policy for a flow on each network hop.
- **Function Configuration:** adaptation of the network functionality, e.g., firewalls, NATs, proxies, through software virtual network functions or programmable bare metal switches.
- **Parameter Configuration:** adaptation of the values and policies to be used by each network function, while flow path and network functionality remain the same.
- **Function Placement:** adaptation of the possible locations for network functions, which has a direct impact on the network performance.
- **Resource and Function Scaling:** adaptation of the resources' assignment to flows or functions, e.g., scaling up processing or storage capacity of a network function, and the adaptation of the number of deployed instances.
- **Topology Adaptation:** adaptation of the graph structure through adding or removing nodes or links, which is provided by virtual networks for instance.

The above aspects can be extended through new networking concepts, technologies or future design requirements.

TABLE II: Cost elements of flexibility.

Resources (CAPEX)	Operation (OPEX)	Adaptation/Migration	SLA
resource overhead	control, data plane throughput	synchronization overhead	fines
network complexity	control, data plane latency	configuration latency	flow interruption
software complexity	energy consumption	topology adaptation latency	network interruption

C. Cost of Flexibility

The price to pay for flexibility in networks is related to extra resources or delays 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 itself such as migration overhead have 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. Intuitively, one might think that costs rise with increased flexibility. However, we lack quantitative evaluation results. Would costs rise linearly, logarithmically or exponentially? Or could a more flexible design lead to cost reduction on the longer term? For a quantitative analysis and, in particular, for the comparison of different design choices, we need to consider all parameters that are related to the flexibility measure and cost.

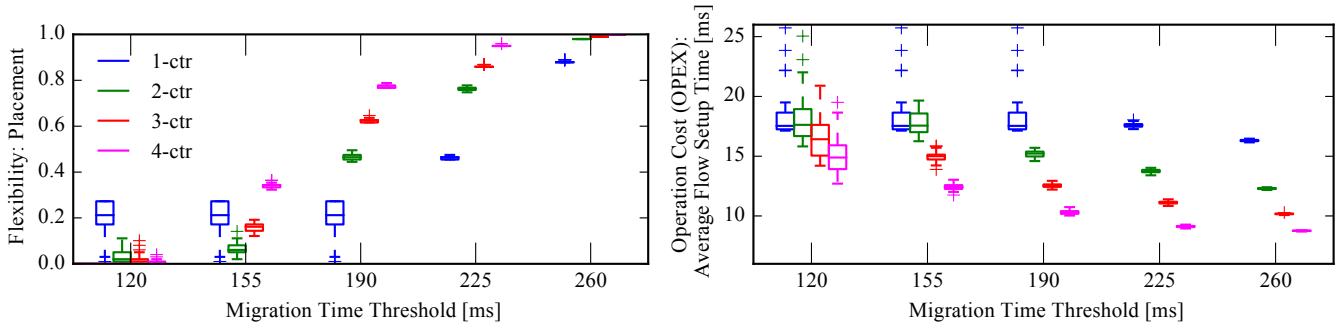
We can distinguish four main categories of cost in relation with flexibility as illustrated in Table II. To provide more flexibility, additional resources might be required (CAPEX), e.g., data centers, and the respective systems might employ a higher complexity to support adaptation, which also translates 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 (category 3), e.g., synchronization overhead or configuration latency. To be able to investigate those separately, we explicitly divide them in two categories. Finally, costs might occur due to the vulneration of SLAs due to adaptation hence change in a network.

TABLE III: Evaluation of network flexibility for selected case studies.

Case Study	Flexibility Aspect	New Request	Flexibility Measure	System Objective	Cost in focus
dynamic SDN controller placement	function placement	new flow arrival (from distribution)	fraction of successful controller placements	control performance: (min. avg. flow setup time)	operation latency (OPEX): avg. flow setup time
failure recovery in SDN (protection vs. restoration)	flow configuration	all possible single and dual failures	fraction of recoverable failures	system recovery: (single and dual failures)	resources overhead (CAPEX): node and link reservation

IV. Case Studies

We demonstrate the use of the above motivated flexibility measure with two selected case studies, which are summarized in Table III. We also hint at design guidelines that can be derived, whereas more case studies are needed to come up with generalized guidelines. The first case study targets the flexibility aspect of placement for SDN controllers. Intuitively, we can expect a higher flexibility with a higher number of controllers as a general design guideline. However, our case study also reveals non-obvious influences of the migration time threshold. The second case study targets the flexibility aspect of flow configuration for a resilience scenario in an SDN network. We compare between protection and restoration for all possible single and dual link failures. One might expect that restoration, due to its reactiveness, is more flexible than a "static" protection approach. However, a failure comprehensive protection approach can also be flexible with respect to failure handling at the cost of protection reservations. Hence, a general design guideline benefits significantly from flexibility analysis.



(a) Flexibility measure in terms of successful control plane migration.

(b) Operation cost (OPEX) in terms of the average flow setup time.

Fig. 2: Flexibility measure and cost comparing different number of controllers systems while varying migration time threshold T of topology Abilene [9].

A. Dynamic SDN Controller Placement

A dynamic SDN control plane can adapt its configuration to changing traffic flows to achieve an optimal control performance. In our case the performance metric is the average flow setup time, i.e., the time involved to detect a new flow by the controller and to setup the flow rules on all switches on its path.

The flexibility of the dynamic SDN control plane represents its ability to respond to changing traffic flows under a migration time threshold T. The configuration, i.e., placement, of the control plane consists of both controller(s) migration and switch(es)-to-controller reassignment. In principle, the controller migration can be implemented through migrating the virtual machine in which the controller application is running. Switch reassignment can be achieved by migrating the control path from the switch to a new assigned controller. The new controller's address is added to the switches while moving the control state from the old to the new controller. Previous studies illustrate that virtual machine migration [10] and control path migration [11] can introduce latency and even control messages loss. Hence, we need to bound the migration time by a threshold T, which depends on the provided network service.

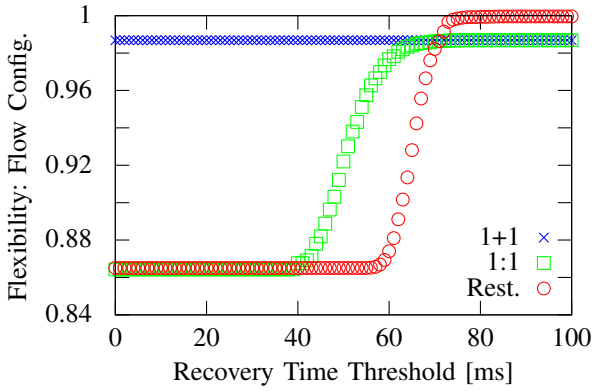
In the evaluation, a new request represents a change in the traffic flows, which are generated randomly to simulate traffic dynamics. A new control plane configuration is not supported if the total amount of time spent in system reconfiguration is larger than T. A new control plane configuration is supported if the new optimal placement is exactly the same as the old one (no adaptation necessary) or if a system reconfiguration is necessary and it finishes within T. An evaluation run is composed of 100 flow requests. See [9] for more details.

We evaluate our flexibility definition comparing four systems that differ in the number of controllers (one to four controllers). As shown in Fig. 2a, for Abilene network topology, a general trend can be observed that the flexibility increases with increasing T, no matter how many controllers are deployed. This follows our intuition that larger T ensures that more reconfigurations can be fulfilled. We can also observe that for smaller T, the flexibility exhibited by one controller (1-ctr) is higher than that of more controllers. This is because there is a high chance that the single controller does not need to change its configuration to achieve its optimal performance. In case of more controllers, the system tends to result in new optimal placements, that require more reconfigurations to be done within T. Hence, only at larger T, the flexibility measure increases with the number of controllers.

Next, we evaluate the systems' operation cost in terms of average flow setup latency while varying T. Generally, higher cost emerges if T cannot be fulfilled by the migration process, where the new optimal placement will be discarded and the average flow setup time will be degraded. Fig. 2b shows that with more controllers available, the system cost decreases as flexibility increases with larger T. For example, the average flow setup time of 4-ctr drops from 15 ms to less than 10 ms,

which is a 30% decrease in cost. In contrast, for 1-ctr case, even if the flexibility increases with larger T up to almost 90% of the requests, the cost improvement is marginal.

For initial design guidelines, the evaluation reveals that the migration time threshold has an important influence on the systems' flexibility and cost which in turn affects our decision for the number of controllers to deploy. Given a stringent T, it is preferred to have only one controller as more controllers do not increase flexibility. More controllers are beneficial under a more relaxed T as they provide high flexibility and improve the cost. The results also show different relationships and trade-offs between flexibility and cost depending on the system.



(a) Flexibility in terms of covered single and dual link failures.

	Resources Cost (CAPEX)	
	Node reservation: Avg. number of flow table entries	Link reservation: Number of required links
$1+1$	11.78	13038
$1:1$	11.78	13038
Rest.	5.05	5400

(b) System resources cost (CAPEX) in terms of nodes and links used for reservation.

Fig. 3: Flexibility measure and cost comparing SDN protection and restoration approaches while varying recovery time threshold T for the COST 266 European reference network (37 nodes and 57 edges) [12].

B. Failure Recovery in Software Defined Networks

The flexibility of survivable networks can be defined as the ability to recover from failures in a timely manner. There have been several proposals for failure recovery in SDN as surveyed in [13]. Network operators can choose either proactive recovery approaches (i.e., protection) where the network is prepared for the failure in advance for the cost of redundant resources; or reactive approaches (i.e., restoration). To avoid network interruption, the reconfiguration of the network after a failure should be performed promptly before the applications are impacted. To help network operators to choose the appropriate recovery approach, we introduce a quantitative measure to compare the flexibility of SDN protection and restoration to recover from all possible single or dual link failures under a recovery time threshold T.

Similar to legacy networks, in SDN the recovery time – the time elapsed between a failure occurs and all switches are adapted to the new network conditions – consists of the following parts [14].

Failure detection: the failure is detected at the responsible network devices, e.g., at neighboring switches. Failure notification: after the failure is localized, the controller is notified, and the network view is updated. Device configuration: an adaptation plan is designed and the new flow rules are installed.

The recovery time of 1+1 protection is quasi instantaneous as this approach duplicates resources (i.e., bandwidth and flow rules) in the form of a disjoint path-pair and uses them simultaneously. Similarly, 1:1 protection reserves backup resources in advance. However, failure detection and notification are required to re-route the traffic from the failed primary to the backup path, which contribute to the recovery time. For restoration, no protection resources are planned and reserved until a failure event occurs. Hence, for restoration, the after-failure tasks are failure detection, notification of the controller, recovery path computation and deployment of the modified flow rules at every switch along the new path, which obviously add more to the recovery time. For evaluation, we assumed that failure detection time is 50 ms using loss-of-signal detection [15]. Propagation delay of the failure notification and configuration messages is calculated with 5 μs per km of optical fiber link, depending on the distance between the controller and the corresponding switches.

The flexibility of the three approaches is depicted in Fig. 3a. The curves show the ratio of recoverable connections for all possible single and dual link failures, while varying T. One can observe that the additional signaling delay makes restoration less flexible for recovery time up to 50 ms, which is usually required in carrier-grade networks. However, if the recovery time threshold is above 70 ms, restoration becomes the most flexible choice, surpassing the others, as it can recover more connections and accommodate more, practically all, dual failures. On the other hand, 1+1 protection shows very high flexibility regardless of T. This is because it provides immediate connectivity after an arbitrary single failure or after a dual failure which affects only the primary path owing to the simultaneous primary and backup flows. However, 1+1 protection is not flexible to cover failures affecting both the primary and backup paths.

For the cost of the evaluated approaches, we analyzed the resources cost in terms of node reservation, i.e., average number of pre-installed flow rules, and link reservation, i.e., number of required links, as shown in Fig. 3b. Both protection approaches

indeed impose the highest cost in terms of flow entries and required links due to reservations needed for the primary and backup paths. Intuitively, restoration is observed to be the most cost effective solution. It achieves cost savings of up to 58% in terms of both node and link reservation.

In general, we observe several broad guidelines comparing the flexibility and cost of failure recovery in SDN. It is clear that 1+1 protection results in the highest system cost, however, it offers high flexibility independent of the recovery time threshold T. Intuitively, restoration imposes the least cost, however, it is only flexible starting from a more relaxed T. Note that restoration is the only approach that could reach full flexibility.

V. Conclusions and Future Directions

Flexibility has emerged as an important design criteria for softwarized networks, but a clear definition of network flexibility is missing. We strive for a definition of network flexibility and propose a quantitative measure to compare different network designs. To illustrate the benefits of a quantitative flexibility analysis, we present two case studies. Dynamic SDN controller placement shows how the time allowed for re-placement influences the observed flexibility of a network design. In the resilience case study, the trade-off between flexibility and cost is highlighted. With this paper, we would like to stimulate the discussion of the use of flexibility as a measure. Further work is needed to come up with a common definition, derive design guidelines and develop an evaluation framework.

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 "Quantifying Flexibility for Communication Networks").

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] ONF, "OpenFlow Switch Specifications 1.5.1," 2015, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf [Accessed: 23.06.2017].

[3] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," *ACM SIGCOMM HotSDN Workshop*, pp. 19–24, 2012.

[4] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," 2014, http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf [Accessed: 23.06.2017].

[5] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[6] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[7] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, "Live migration of an entire network (and its hosts)," *ACM HotNets Workshop*, pp. 109–114, 2012.

[8] W. Kellerer, A. Basta, and A. Blenk, "Using a Flexibility Measure for Network Design Space Analysis of SDN and NFV," *Proceedings of INFOCOM workshop SWFAN*, 2016.

[9] M. He, A. Basta, A. Blenk, and W. Kellerer, "How Flexible is Dynamic SDN Control Plane?" *Proceedings of INFOCOM workshop SWFAN*, 2017.

[10] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. De Laat, J. Mambretti, I. Monga, B. Van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, 2006.

[11] A. Basta, A. Blenk, H. B. Hassine, and W. Kellerer, "Towards a dynamic SDN virtualization layer: Control path migration protoco," *IEEE Conference on Network and Service Management (CNSM)*, pp. 354–359, 2015.

[12] COST 266 European reference network, http://sndlib.zib.de [Accessed: 23.06.2017].

[13] A. S. da Silva, P. Smith, A. Mauthe, and A. Schaeffer-Filho, "Resilience support in software-defined networking: A survey," *Computer Networks*, vol. 92, pp. 189–207, 2015.

[14] J. Tapolcai, P.-H. Ho, P. Babarczi, and L. Rónyai, *Internet Optical Infrastructure - Issues on Monitoring and Failure Restoration*. Springer, 2015.

[15] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "In-band control, queuing, and failure recovery functionalities for OpenFlow," *IEEE Network*, vol. 30, no. 1, pp. 106–112, 2016.