Overcoming Network and Security Management Platform Gaps in Federated Software Networks

1st Michael Steinke
Bundeswehr University Munich
Neubiberg, Germany
michael.steinke@unibw.de

2nd Wolfgang Hommel
Bundeswehr University Munich
Neubiberg, Germany
wolfgang.hommel@unibw.de

***Abstract*—Software networks are on the verge of replacing traditional communications network infrastructures on a large scale; they are composed of software-defined networks, virtualized network functions, and virtual compute resources. While software networks thrive on scalability, high dynamics, and flexibility, they also are inherently tied to conceptually new challenges for network management, i. e., the tasks carried out by network operators in order to provision, maintain, and optimize network-based IT services. While individual management tools are made available for each software network component by the vendors, integrated management architectures, which enable the common management of components across the heterogeneity of vendors and types as well as models, need to be redesigned to achieve the same level of maturity and usefulness compared to their counterparts in traditional networks.**

This paper focuses on management platforms for federated software networks – operated by several independent parties – systematically, based on their key characteristics. We identify gaps in management platform design and propose measures to close them.

***Index Terms*—network management, security management, management architecture, SDN, NFV**

I. Motivation

Software networks (SNs) constitute the current state-of-the-art in networking, which resulted from the transition from traditional hardware-based networks (HNs) to software-based ones [24]. Often solely used as an interchangeable term for virtual networks, SNs encompass a by far wider meaning: SNs describe a fundamental redesign of the mode of operation, compared to conventional approaches; they are *at least* characterized by the following aspects:

- **C.1** SNs are established *atop a shared platform*, often spanning over multiple {*physical*, *virtual*} nodes (often SNs themselves), almost *entirely decoupled* from the platform's architecture and topology.
- **C.2** Instead of having a configuration in situ, as many hardware-based networking devices require, SNs can be deployed, configured, and maintained in a *remote*, *centralized* manner.
- **C.3** SNs allow a seamless operation of *geographically distributed* networks, leveraging *federated infrastructures*.
- **C.4** The configuration of SNs can be adjusted *immediately* (within seconds to minutes) instead of hours or days.
- **C.5** From a management point of view, IT resources are no longer encapsulated in individual servers; servers provide their IT resources to a *common pool of shared resources*.
- **C.6** IT resources can be assembled, deployed, and used as required in a *scalable* manner.
- **C.7** SNs' *dimensions* are often far more extensive than HNs'.
- **C.8** SNs are often composed of a *heterogeneous* set of interoperating technologies (e. g., SDN, NFV, virtualization), management and orchestration (MANO) systems, and components by different vendors.

Federated software networks (FSNs) span over several, often geographically widely distributed data centers owned and operated by multiple parties, which *share a common or complementary objective* and IT resources stipulated by an IT service contract. FSNs have *at least* these extra characteristics:

- **C.9** FSNs' operation, administration, and use is based upon *trust* between multiple *more or less familiar parties*.
- **C.10** Operation of applications and services on infrastructure with multiple *intersecting administrative domains*.
- **C.11** *Various management concepts* for shared IT infrastructures, including organizational aspects (e. g., structure and responsibilities), data models, tools, and interfaces.
- **C.12** *Organization-wide management decision-making* (e. g., w. r. t. software changes in management platforms, organization, etc.) with side-affects to federation partners.
- **C.13** *Segments* of IT resources are *operated and managed by multiple tenants*.

SNs do leverage federated infrastructures, e. g., allowing organizations to purchase and integrate virtual IT resources on demand from different providers into their own infrastructure, each along with a proprietary management interface.

Management is *the* essential task to provide services' reliability, security, and efficiency. It comprises various functional areas (e. g., *FCAPS* [11]), concerning activities, tools, and methods in favor of operating, administrating, maintaining, and providing networked systems [9]. Security management intends to preserve confidentiality, integrity, and availability of information and IT resources [12]. Despite its importance, it is often disregarded and limited to preventive measures and alerting (cf. ISO management framework [11]). Network and security management (NSM) pursues an *integrated* approach

978-3-903176-14-0

of both fields, given interrelations of information, causation, and effects (e. g., a service failure due to malicious activities).

FSNs are inherently not managed from one central instance (e. g., an ISP) due to divergent interests or needs of the independent federation partners; networks and IT resources are logically isolated for each tenant respectively – they must be managed individually *and* in its entirety. Furthermore, a lot of present management platforms cannot cope with FSNs. They are typically *a)* very limited w. r. t. their field of application (i. e., use cases and systems), or they provide *b)* multi-purpose NSM, yet with insufficient design for FSNs and limiting FSNs' potentials. A NSM platform, adapted to FSNs, is needed to holistically reap their benefits and leverage versatile use cases.

This paper exposes *open gaps in present management platforms* for FSNs based on their characteristics; we first examine existing platforms w. r. t. their operability in the field of FSNs in Section II. We then *address the gap* between NSM platforms in their current state and a desired state for managing FSNs in Section III and *propose measures towards a gap closure* in Section IV. Section V summarizes key findings and outlines our future work to overcome the gaps.

II. NSM in existing Platforms

FSNs can be composed in various ways. Often, they are enabling subsystems for additional applications, e. g., in cloud-computing. This analysis considers a *cross section* of the most extensive existing approaches, grouped by their corresponding paradigms and *shows present drawbacks in NSM platforms*.

A. Cloud Computing

OpenStack (OS) and *OpenNebula* (ONE) are both similar platforms for (F)SNs. They aggregate geographically distributed IT resources (*C.3*) into a shared pool (*C.5*), a remote, centralized (*C.2*), and instant (*C.4*) configuration nested (*C.1*) SNs. They integrate various paradigms (e. g., virtual network functions and arbitrary SDN controllers), providing heterogeneous (*C.8*) systems. OS does consider the need of resource and identity [4] federations, especially in a consumer-to-provider relation [5], yet, regarding the latter one, without a detailed description. ONE has similar functions based on the master/slave model [23]. A federation with other platforms from other vendors is not supported. Both lack typical NSM functions, especially for {performance, security} management.

B. Federated Cloud Platforms

The BEACON framework [16] is one of the few cross-system platforms for cloud federations, fitting different federation models [17]. It also integrates NSM tasks with focus on intrusion detection, vulnerability scanning or DDoS and proposes a set of security aspects for {application, infrastructure} security. A key feature of BEACON is the configuration of overlay networks, i. e., the placement of virtual resources and services over geographically distributed sites. The implementation, however, is limited to ONE, OS and OpenDaylight for network management (cropping heterogeneity (*C.8*)). FSNs' characteristics *C.9*, *C.11*, *C.12* are not considered.

C. Software-Defined Networking

OpenDaylight (ODL) and the *Open Network Operating System* (ONOS) claim to allow monitoring and controlling of network devices [21] [14]. The key benefit of ODL is its extensibility and adaptability via the YANG data modeling language [8], allowing to handle FSNs' heterogeneity (*C.8*). It is designed for a separation of platform and network functionality (*C.1*), and a remote, centralized (*C.2*), immediate (*C.4*) configuration, yet, with a too limited scope on particularly OpenFlow-capable components. ODL can be operated in a federated cluster [20], providing huge (*C.7*) as well as geographically scattered SNs (*C.3*), however disregarding federated NSM. NSM functions are partially implemented (e. g., [25] [19]), considering multiple tenants [22] (*C.13*). An AAA service furthermore allows the usage of LDAP services for user management, but is limited in its implementation [18]. Like ODL, ONOS serves as a platform for Apache Karaf applications. Holistic NSM functions are not available; however, an application for SNMP-based alarm aggregation as well as a traffic analysis and monitoring application can be added [13]. A federation of multiple (ONOS) clusters is considered to be an orchestration level task and not provided by ONOS [10]. ODL and ONOS implement NSM functions in parts, yet lack a holistic integration.

D. Service Orchestration and Management

Software containers rendered a lightweight alternative to hardware virtualization, forming SNs themselves with several yet established management platforms, like *Kubernetes* (K8s) as one of the most sophisticated platforms. K8s provides an API for network and system configuration of containers and service composition via "Pods", and basic role-, label-, or parameter-based policy descriptions to define network access restrictions [27]. It does not provide FSNs nor management across multiple organizations; however, K8s provides the federation of clusters of nodes. A limited concept similar to administration domains is realized via namespaces [29], allowing the separation of multiple teams. Add-ons like *Nuage Network VCS* allow security monitoring [28]. Similar systems like Docker Swarm, Apache Mesos, Marathon, and Nomad lack a holistic NSM with required functions. None of the analyzed systems does, however, provide appropriate functionality for network and service federation, nor federated management.

E. General-purpose Management and Automation Platforms

One of the most extensive management platforms with focus on network monitoring is *OpenNMS*. Yet, it cannot handle the characteristics of FSNs in crucial aspects: {Network, device} discovery must be implemented manually or periodically via ICMP [1], leading to inconsistent views of the network status. It generally lacks protocol implementations, interfaces, and connectors to be integrable in software networks, which, however, could be expanded via modules. OpenNMS allows the definition of critical paths, which may be used for the mapping of platforms to overlying FSNs. Although OpenNMS provides user management abilities, multi-tenancy is not given:

Platform	Fault	Configuration	Accounting	Performance	Security
OpenStack	✓	✓	✓	✓	✗
OpenNebula	✗	✓	✓	✓	✗
BEACON	●	✓	✓	✓	●
OpenDaylight & ONOS	✗	●	✗	✗	✗
Kubernetes	✗	✓	●	✗	✗
OpenNMS	✓	●	●	✗	●
ONAP	✓	✓	✓	✓	✓

Table I: FCAPS in existing platforms. (✓: fit; ●: partial fit; ✗: unfit)

It allows a superficial concealing of managed objects according to their categories, yet, users may still access their data [7]. *Open Network Automation Platform* (ONAP) [2], a more recent approach, facilitates the design, creation, orchestration, monitoring, and life cycle management on the basis of a policy-driven approach [15]. Besides an *execution framework*, it provides a *design-time framework*, allowing operators the design of services (a combination of IT resources) and policies. ONAP is inherently planned to provide all management functions (FCAPS) described by the ISO [2]. Since ONAP is designed as automation platform for software-based networks, its design largely considers characteristics of SNs (*C.1–C.8*). ONAP allows the communication to network control platforms like SDN controllers and VM and service deployment platforms like OpenStack, other 3rd party controllers, and services like identity and access management (IAM) information [3]. Though, the documentation does not describe the integration of information and functions from network and system management tools in general. In addition, ONAP is not designed to provide a shared platform for multiple tenants in a *federated* software-based network, managed by multiple autonomous organizations: The relationship between multiple cooperating organizations is not sufficiently considered (*C.9*), although ONAP inherently integrates access control mechanisms [6] to protect informations and functionality. A more fine-grained and flexible approach is needed (e. g., determination of shared and isolated resources like services and information). Furthermore, it does not provide adequate answers to multiple, intersecting domains in FSNs (*C.10*), to versatile management concepts from diverse federation partners (*C.11*), and to effects of local (organization-wide) decision for federation partners (*C.12*). Also, important information w. r. t. multi-tenancy aspects is missing – e. g., how ONAP may provide a tenant's view across multiple platforms or handles and processes events and policies securely in networks managed by multiple tenants.

III. The Gap in NSM platforms for Federated Software Networks

An overview of FCAPS capabilities of management platforms in FSNs is shown in Table I. ONAP explicitly considers all functional areas. Other approaches vary in their functional range but usually lack at least one. In detail, several open gaps are especially noticeable:

G.1 A ubiquitous lack of **cross-platform NSM capabilities** or common usable ways of cross-platform integration (e. g., expandable connectors to other systems).

G.2 Incompatibility of **existing infrastructure and tools** like from IAM (e. g., LDAP-based), configuration management (e. g., a CMDB), or variable monitoring as well as network operation systems.

G.3 Non-integrability of **existing information**, e. g., about users, federated organizations, responsibilities, monitoring events, etc. due to inflexible data formats.

G.4 Disregard of **NSM chances, enabled by software networks** – e. g., a preventive control and *a priori* alignment of conditions from configuration changes with policies.

G.5 Disregard of components from **underlying HNs** as crucial part of the infrastructure.

G.6 A lack of **integrated NSM functions** in FSNs. Functions, for instance, lack holistic component and service discovery for an overall management view completely or by using obsolete interfaces/protocols (e. g., SNMP).

G.7 A lack of adequate data models, processes, and functions for **multi-tenancy** abilities.

G.8 A lack of **integration of NSM functions** (cf. Table I).

G.9 A lack of an adequate platform-integrated ability to describe the **desired state** (e. g., by *policies*) of FSNs, in order to automatically detect incidents and problems.

G.10 A lack of **consolidation** of *a)* FSNs' desired state, *b)* their actual state (monitoring information), and *c)* countermeasures to recover the FSNs' desired state.

G.11 A lack of mechanisms to **isolate federated organizations** in an environment (i. e., to guarantee a secure operation of the shared infrastructure and information).

G.12 A mis-design of **domain-models**; especially a holistic consideration of *integrating existing dependencies* between resources and organizational information in a federated environment.

G.13 The inability to **integrate different management concepts** from diverse federation partners, and the inadaptability of theorganizational structure (e. g., groups, rights, and implicated responsibilities).

G.14 Disregard of a **decentralized decision-making** and their affects to the network.

An alignment of the shortcomings listed above with respect to concerned characteristics of MANO platforms for FSNs is summarized in Table II. The major gap is particularly distinctive to the realization of a federated, integrated network management in already existing as well as subsequently expanded *heterogeneous* infrastructures. Both cases do represent viable scenarios and must be fostered in order to benefit from FSNs in practice. Consider, e. g., the following scenarios:

(1) A cooperation of several research groups, establishing a shared virtual network built atop, yet segregated from, existing infrastructure. (2) An organization planning to seamlessly expand its infrastructure with resources from an external service provider for a confined period of time.

IV. Platform Architecture Designs towards closing the identified Gaps

An overview of design elements, which contribute to a gap closure and may consequently leverage the use of FSNs in

	Suitability in SNs								Suitability in Federated SNs				
Platform	*C.1*	*C.2*	*C.3*	*C.4*	*C.5*	*C.6*	*C.7*	*C.8*	*C.9*	*C.10*	*C.11*	*C.12*	*C.13*
OpenStack	✓	✓	✓	● **G.4**	✓	● **G.6**	✓	✗ **G.1**,**G.2**,**G.3**	✗	●	✗	✗	✓
OpenNebula	✓	✓	✓	● **G.4**	✓	● **G.6**	✗ centralized	✗ **G.1**,**G.2**,**G.3**	✗	●	✗	✗	✓
BEACON	✓	✓	✓	● **G.4**	✓	● **G.6**	✓	● OS + ONE + ODL	✗	●	✗	✗	✓
OpenDaylight & ONOS	● **G.1**,**G.5**	● **G.6**	✓	✓	✓	✗ **G.4**	✓	✓ expandable	✗	✗	✗	✗	✗ **G.6**,**G.7**
Kubernetes	✓	✓	✓	● **G.4**	✓	✓	✓	✗ **G.1**,**G.2**,**G.3**	✗	●	✗	✗	✓
OpenNMS	✓	✗ **G.6**	✓	✗ **G.6**	✗	✗ **G.6**	✗ **G.6**	● **G.1**,**G.6**	✗	✗	✗	✗	●
ONAP	✓	✓	✓	✓	✓	✓	✓	● **G.2**	● access controls	✗	✗	✗	●

Table II: Overview of NSM compliance with characteristics in FSNs. (✓: fit; ●: partial fit; ✗: unfit)

productive environments, is shown in Table III. In total, we identified 17 design elements from three different groups: *Data modeling*, *architecture design*, and *functional scope*:

Data modeling includes measure **M.1**. An adapted *internal information meta-model* is needed (e. g., based on [26]) to provide an adequate abstraction of FSNs' extremely nonuniform, heterogeneous environment (e. g., mapping identical functions from different systems, IT resources, users, responsibilities, groups, etc.). A well designed meta-model addresses practically all gaps.

Architecture design encompasses measures **M.2**–**M.5**, **M.9**–**M.11** and **M.14**–**M.16**. The main focus of this category is on the NSM platform's ability to be adaptable to the managed environment – we think that this is one of the *most important requirements* for management platforms in FSNs. This includes the ability to allow quick adaptations to MANO APIs, a logical centralized processing (yet architectural decentralization to meet performance requirements), the ability to act as a proxy between users and MANO platforms, allowing preventive measures (e. g., by checking user input for policy violations). The whole platform must, in fact, be configurable, w. r. t., its functional range, responsibilities, policies, NSM functions, privacy concerns, type of collaboration between management domains, and an intra-management-domain organization. Hence, different management concepts from all federation partners can be considered in a shared platform.

The remaining measures can be assigned to measures in order to improve NSM platforms' *functional range*. Monitoring must be expandable by a correlation functionality across information from all NSM functions (e. g., performance issues considering security problems). However, management platforms, must also be capable to configure (control) FSNs. Furthermore, a precise, up-to-date documentation of the environment is important in order to have an overview of instances, dependencies, problems etc.

Measure ID	Description	Gaps
M.1	Comprehensive meta-model for management functions and information	**G.1**–**G.14**
M.2	Formalization of MANO API descriptors	**G.1** **G.2** **G.4** **G.5**
M.3	Distributed architecture of the platform	**G.4**
M.4	Centralized evaluation and handling	**G.4**
M.5	NSM platform as proxy between users and MANO systems	**G.4**
M.6	Capability of platforms to reconfigure FSNs	**G.4**
M.7	Event correlation across all NSM functions	**G.6** **G.8**
M.8	Transfer of findings across all NSM functions	**G.8**
M.9	Expandability of the NSM platform's functions and procedures	**G.6**
M.10	Configurability of responsibilities	**G.7**
M.11	Adaptivity of policies' scope	**G.9**
M.12	Adequate abstraction of policies	**G.9**
M.13	Completeness of NSM processes	**G.10**
M.14	Configurable privacy per partners	**G.11**
M.15	Collaboration between Management Domains	**G.12**
M.16	Adaptable intra-domain Organization	**G.13**
M.17	Network Documentation and Dependencies	**G.14**

Table III: Measures to close identified gaps in NSM platforms in FSNs (Grouping: Data model , architecture design , functional scope).

V. Conclusion and Outlook

In this paper we first stated the differences between FSNs, SNs and traditional HNs by specifying the key characteristics of the former two. Focussing on FSNs, we summarized the required functionality for network and security management. By analyzing the management capabilities of existing platforms from various areas in the field of (F)SNs, we identified gaps between the management functionality *required* to meet the characteristics of software networks and *present* functionality in current implementations.

While each management tool has its individual strengths and shortcomings, we proposed a common set of measures to close remaining gaps. Our proposal shows that especially gap **G.4**, the alignment of configuration changes with policies, requires a series of individual actions; in contrast, improvements to a meta-model (**M.1**) yield quick wins in several areas.

Our own future research focuses on information modelling (**M.1**), and integrated management architectures to provide integrability of FSNs in existing infrastructures (**M.5**, **M.8**, **M.9**, **M.14** – **M.17**), with the goal of creating a software framework as basis for fit management platforms in FSNs.

Acknowledgement This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1); it is partly funded by the German BMBF (Project Id 16KIS0549).

REFERENCES

[1] OpenNMS Administrators Guide, June 2015. https://docs.opennms.org/opennms/releases/16.0.2/guide-admin.

[2] ONAP Architecture Overview (Open Network Automation Platform (ONAP) Architecture White Paper), 2017. https://www.onap.org/wp-content/uploads/sites/20/2017/12/ONAP_CaseSolution_Architecture_120817_FNL.pdf.

[3] ONAP Security Framework, Jan. 2017. https://wiki.onap.org/display/DW/Security+Framework.

[4] Federated keystone, May 2018. https://docs.openstack.org/security-guide/identity/federated-keystone.html.

[5] Inter cloud resource federation, June 2018. https://wiki.openstack.org/wiki/Inter_Cloud_Resource_Federation.

[6] ONAP Application Authorization Framework, Jan. 2018. https://wiki.onap.org/pages/viewpage.action?pageId=3247151.

[7] User Restriction Filters, Mar. 2018. https://wiki.opennms.org/wiki/User_Restriction_Filters.

[8] M. Bjorklund. Yang – a data modeling language for the network configuration protocol (netconf), Oct. 2010. https://tools.ietf.org/html/rfc6020.

[9] A. Clemm. *Network management fundamentals*. Cisco Press, 2006.

[10] David Boswell, Uyen Chau. Roadmap, Apr. 2017. https://wiki.onosproject.org/display/ONOS/Roadmap.

[11] International Organization for Standardization and International Electrotechnical Commission. Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework, Nov. 1989.

[12] International Organization for Standardization and International Electrotechnical Commission. *Information technology - Security techniques - Information security management systems - Overview and vocabulary*. International Organization for Standardization, International Electrotechnical Commission, Feb. 2016.

[13] P. Joshi and A. Danoush. New Projects, Dec. 2017. https://wiki.onosproject.org/display/ONOS/New+Projects.

[14] Linux Foundation Administrators. Wiki Home, Sept. 2017. https://wiki.onosproject.org/.

[15] K. McDonnell. What is ONAP?, Oct. 2017. https://wiki.onap.org/pages/viewpage.action?pageId=1015843.

[16] R. Moreno-Vozmediano, E. Huedo, I. M. Llorente, R. S. Montero, P. Massonet, M. Villari, G. Merlino, A. Celesti, A. Levin, L. Schour, et al. Beacon: A cloud network federation framework. In *European Conference on Service-Oriented and Cloud Computing*, pages 325–337. Springer, 2015.

[17] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45(12):65–72, 2012.

[18] OpenDaylight Project. Authentication, Authorization and Accounting (AAA) Services, 2016. http://docs.opendaylight.org/en/stable-carbon/user-guide/authentication-and-authorization-services.html.

[19] OpenDaylight Project. Centinel user guide, 2016. http://docs.opendaylight.org/en/stable-nitrogen/user-guide/centinel-user-guide.html.

[20] OpenDaylight Project. Federation:main, 2016. https://wiki.opendaylight.org/view/Federation:Main.

[21] OpenDaylight Project. Introduction. http://docs.opendaylight.org/en/stable-nitrogen/getting-started-guide, 2016.

[22] OpenDaylight Project. Virtual tenant network (vtn), 2016. http://docs.opendaylight.org/en/stable-nitrogen/user-guide/virtual-tenant-network-(vtn).html.

[23] OpenNebula Project. Opennebula federation configuration, 2018. https://docs.opennebula.org/5.4/advanced_components/data_center_federation/federationconfig.html.

[24] G. Pujolle. *Software Networks: Virtualization, SDN, 5G, Security*. John Wiley & Sons, 2015.

[25] R. Srivastava and G. Pande. Cardinal: Opendaylight monitoring as a service, Jan. 2018. https://wiki.opendaylight.org/images/8/89/Cardinal-ODL_Monitoring_as_a_Service_V2.pdf.

[26] M. Steinke and W. Hommel. A data model for federated network and security management information exchange in inter-organizational it service infrastructures. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–2. IEEE, 2018.

[27] The Kubernetes Authors. API Overview, 2016. https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.9/.

[28] The Kubernetes Authors. Cluster Networking, 2018. https://kubernetes.io/docs/concepts/cluster-administration/networking/.

[29] The Kubernetes Authors. Namespaces, 2018. https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/.